A MULTILINGUAL SCREEN READER IN INDIAN LANGUAGES

E.Veera Raghavendra†, *Kishore Prahallad*†‡

†International Institute of Information Technology - Hyderabad, India.
‡Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA.
raghavendra@iiit.ac.in, skishore@iiit.ac.in

ABSTRACT

Screen reader is a form of assistive technology to help visually impaired people to use or access the computer and Internet. So far, it has remained expensive and within the domain of English (and some foreign) language computing. For Indian languages this development is limited by: availability of Text-to-Speech (TTS) system in Indian languages, support for reading glyph based font encoded text, Text Normalization for converting non standard words into standard words, supporting multiple languages. In this paper we would discuss how to handle these issues in building multilingual screen reader in Indian languages.

***Index Terms*—** speech synthesis, font identification and screen reader

1. INTRODUCTION

A screen reading software speaks out the contents on the current screen (display) of a computer system. Such software would help a visually challenged user to access the computer and Internet. Typically, screen readers have remained expensive and within the domain of English.

Some of the limitations observed in existing screen readers [1] [2] [3] include: 1) Professional English screen readers fail to provide good support for Indian languages. The voices have US/UK accent and thus native Indian speakers found it hard to comprehend. 2) Screen readers in Indian languages support one or two major speaking languages. At the same time, they often support only Unicode formats and thus ignore the best local websites such as Eenadu, Vaartha, and Jagran, which use ASCII based fonts. 3) Some screen readers do not make use of recent advances in text-to-speech technologies and thus use robotic synthetic voice.

Our mission is to develop a multi-lingual screen reader (for visually impaired) system which can read contents in all official languages of India such as Hindi, Telugu, Tamil etc., including Indian English, and provide support for different computer applications (email, Internet, office software) using intelligible, human-sounding synthetic speech.

The rest of the paper is organized as follows. Section 2 explains the system architecture. Section 3 discusses each component of the system. Section 2.3 explains the implementation of the text-to-speech system.

2. SYSTEM ARCHITECTURE

The system architecture in Figure 1 is modular based and it consists of three sub-systems. They are (i) Text Extractor sub-system, (ii) Text Preprocessing sub-system and (iii) Text-to-Speech systems sub-system. The sub-systems are further divided into many modules as given below.

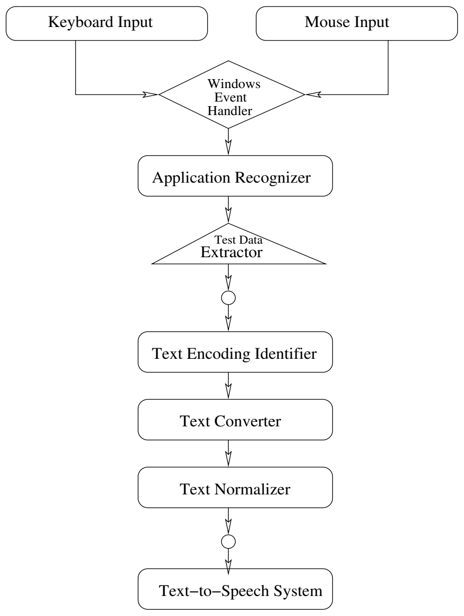

Fig. 1. System Architecture.

2.1. Text Extraction

Text extraction sub-system has functionally classified into: (i) *Event tracer* for capturing the system wide events, (ii) *Keyboard or mouse* input capturer, (iii) *Application recognizer* for identifying events raised by active application or by which object such as button, check box etc., and (iv) *Text extractor* for extracting the relevant text and some text features (especially font name) from the active application or object based upon the current cursor position.

2.2. Text Processing

Text preprocessing sub-system is functionally classified into: (i) *Text encoding or font-type identifier* for determining extracted text is encoded in Unicode or ISCII or ASCII type font, (ii) *Text converter* for converting these different encoded text into a phonetic transliteration notation IT3, (iii) *Text normalizer* for converting the Non-Standard Word (NSW) (like currency, time, title etc.,) into a standard pronounceable/readable words, and (iv) *Language identifier* for identifying the language constituent for selecting appropriate TTS system.

2.3. Text-to-Speech System

Text-to-speech sub-system is functionally classified into: (i) *TTS selector* for switching between different language TTS systems based on language, (ii) *Voice loader* for loading dynamically available voice for current language (iii) *Voice inventory* for maintaining prerecorded speech segments and language related features, (iv) *Unit selection* algorithm for selecting the optimal speech segment for concatenation, and (v) *Speech segment concatenation*. The TTS sub system uses techniques explained in [4], [5], [6]. The voices are built on Festvox framework [7].

3. COMPONENTS OF SCREEN READER

Developing a full-fledged screen reader for Indian languages is not only a challenging but a daunting task considering the eventualities aspects involved in the design and implementation. As we have addressed the common issues in developing a screen reader in the Section 1, we now look into some intrinsic application such as internal text storage, identifying the font type name, identifying the language, converting, normalizing the text as TTS renderable form (IT3 data) and invoking the TTS system which is capable of synthesizing the voice for the identified language text on the screen.

3.1. Internal Storage Format of Text

The text extractor module extracts the relevant text which itself is complicated task for Indian languages. Because of the nature of text as it can be represented in different formats, mark-up (tagged) formats, encrypted, proprietary formats etc. Relevant text extraction means part or position of the text such as paragraphs, sentence, word, cell, title, cursor position, character echoing etc., such fields explicitly restores the text for retrieving in future. Now-a-days many document contains multilingual text in the form of ANSI, True Type Font (TTF) and Unicode text. Based upon the font name and character range we identify the text that belongs the langauge and invoke the corresponding modules to process the text. More over a suitable data type has to be identified to store internally the extracted text. In this regard we faced problem on trying to extract and store Unicode data. Later we found that we should use a suitable data type to retrieve it. So, we identified and used data types like ANSI C string and character pointer for ASCII based text and binary string (bstr) for Unicode data to store internally. Extracting text from Notepad and WordPad is complicated since they don't have any Object Library files. So, we were restricted to use cursor position and Windows messages to extract the relevant text.

3.2. Font Processing

Most of the Indian language electronic content is available in different font encoding formats. Indian language content processing is a real world problem for a researcher. Here processing includes identifying the font encoding and converting the text into TTS renderable input format. To process the font-data first and foremost job of a screen reader is to identify the underlying encoding or font. There will be no header information like in the case of Unicode (UTF-8) encoding or a distinguished ASCII code ranges for English and Indian languages like in ISCII. All we can get is the sequence of glyph code values (ASCII values). So these can be identified through statistical modeling or machine learning techniques. The statistical models are generated by following some statistical methods using the glyph codes. Converters or models are developed using some machine learning algorithm to learn the glyph code order. And these converters or models are capable of identifying the font using the statistical models. The font identification module uses vector space models for font-type identification [8], [9].

3.3. Text Preprocessing

In screen reader perspective text processing means converting the extracted text in some encoded/compatible format for the TTS system, i.e font-to-IT3 conversion, Unicode text conversion, ISCII text conversion etc. Then we need to normalize the converted text. Text normalization is the process of converting non-standard words like numbers, abbreviations, titles, currency etc., to expanded/natural (pronounceable) words. This can be achieved by writing a text Normalizer for each and every language or designing a generic framework which fits for most of the languages. There is one more step which is optional. If we are using some third party TTS systems then we need to convert text in the notation supported by that TTS system, since the input notation for TTS is vendor/developer specific. If we are using our own TTS system then there is no such overhead.

3.4. Language Identification

The language information of the text is very important to invoke the right TTS system. This can be done in two ways either identify the language of the raw text (extracted text) or identify the language from the converted text (phonetic text). Based on the extracted/identified font name it identifies the language, else it identifies the language based on the phonetic sequences.

4. TEXT-TO-SPEECH SYSTEM

A text-to-speech system converts the given text into corresponding spoken form. The current state of art systems are concatenative synthesis [10] and statistical parametric synthesis [11, 12]. The size of unit selection speech synthesis is between few hundred of MBs to GBs. Such a huge database requires a large memory size and slows down the computational speed. It also causes too much hindrance to download and install in ordinary machine. Existing systems uses pruned databases for concatenative synthesis. Hence, the quality of the synthesis is not natural. Currently we are planning to integrate statistical parametric synthesis. The size of such synthesis is very small and easily downloadable. We have built statistical parametric synthesis using Artificial Neural Networks (ANN). Following sub-sections explains in detail.

4.1. Overview of the ANN based synthesis

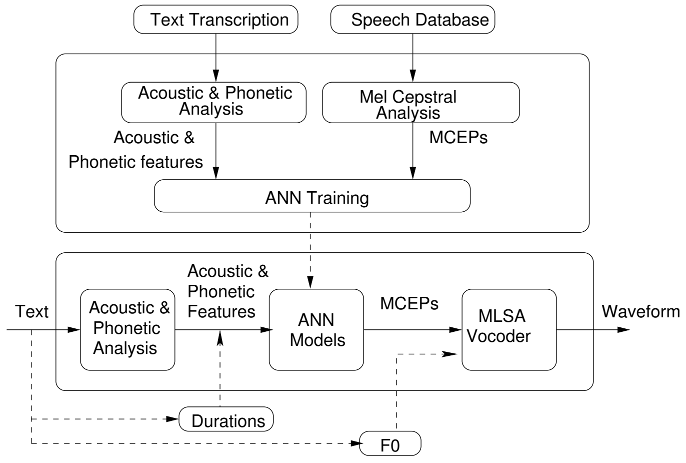

Fig. 2. Mel Cepstral based ANN synthesis architecture

The complete ANN synthesis system is shown in Fig 2. The text-to-speech system includes a text-to-acoustic&phonetic analysis subsystem, one/more neural network models used to predict MCEPs. During synthesis, the given text is converted into acoustic & phonetic notation and MCEPs are predicted using existing models. Here the durations of each phoneme for and fundamental frequencies (f_0) are taken from the test sentence database. Predicted MCEPs and original f_0 are given to MLSA vocoder to synthesize speech.

4.2. Artificial neural network models for speech synthesis

Artificial Neural Network (ANN) models consist of interconnected processing nodes, where each node represents the model of an artificial neuron, and the interconnection between two nodes has a weight associated with it. ANN models with different topologies perform different pattern recognition tasks. For example, a feedforward neural network can be designed to perform the task of pattern mapping, whereas a feedback network could be designed for the task of pattern association. ANN models are also known to capture complex and nonlinear mapping and for their generalization behavior. In the context of speech synthesis, a mapping is required from text (linguistic space) to speech (acoustic space). Thus we exploit the pattern mapping capabilities of ANN models to perform complex and nonlinear mapping of linguistic space to acoustic space to generate synthetic speech.

4.2.1. Input Representation of Text

Since we are performing a mapping from text input space to formant output space, a careful representation is needed at the input layer as such mapping is not only complex but we also expect the ANN model to produce subtle variations in the formants and bandwidths for every frame.

Features extracted from the text to train the ANN model is shown in Table 1. The features include current, left and right phone articulatory and syllable features. Along with these, current phone position in the word, current word position in the sentence and temporal features (position of the frame within the current phone) and state information of the current frame. Please note that the state information is incorporated in the ANN modeling to help to differentiate the frames within a phone. To represent the temporal variations, fifteen time index neurons are used within a state, following formula [13] is used. These time indices represent the relative position of the current frame within a state of a phone segment. This helps to smooth transition between neighboring frames especially on state and segment boundaries. The value of time index i during frame j is calculated using Eq 1 (we have chosen β = 0.01), such that time index i reaches its maximum value during frame $j = i$.

$$O_i = exp(-\beta(i-j)^2) \tag{1}$$

4.2.2. Output Representation

The network is expected to predict Mel Cepstral Coefficients (MCEPs) at the output layer. 25 coefficient vector is predicted for each 10ms frame size with 5ms frame shift. The ANN model is trained for 200 iterations using back propagation learning algorithm.

Table 1. *Input features to predict formants and bandwidths*

(a) Overall features to map between input and output

Feat. Name	# Bits	Rep.
Current phone articulatory features	29	Binary
Previous phone articulatory features	29	Binary
Next phoneme articulatory features	29	Binary
Current phone position in the word	3	Binary
Current phone syllable features	9	Binary
Previous phone syllable features	9	Binary
Next phone syllable features	9	Binary
Current word position in sentence	3	Binary
Temporal features	15	Float
Phone Duration	1	Float
Total	136	

(b) Articulatory features used in the input

Feat. Name	Feat. Values	# Bits
Phone type	Vowel/Consonant	2
Vowel length	Short/long/diphthong/schwa	4
Vowel height	High/middle /low	3
Vowel frontness	Front/mid/back	3
Lip rounding	+/-	2
Consonant type	Stop/fricative/affricative/ nasal/lateral/approximate	6
Place of articulation	Labial/alveolar/palatal/ labio-dental/dental/velar/glottal	7
Consonant voicing	voiced/unvoiced	2

(c) Syllable features used in the input

Syllable Feat.	Feat. Values	# Bits
Stress	True/false	1
Phone type	Onset/coda	2
Phone position in syllable	Begin/middle/end	3
Syllable position in word	Begin/middle/end	3

(d) Other features used in the input

Feature name	Feat. Values	# Bits
Phone position in the word	Begin/middle/end	3
Word position in the sentence	Begin/middle/end	3
Phone state information	Begin/middle/end	3

4.3. Experiments

The purpose of this neural network is to generate MCEPs. Features mentioned in 1(a) are used to represent mapping between input and output. Generally statistical models require huge amount of data to analyze the training patterns. Hence, we wanted to build model with only one network. The architecture of the feedforward network used in this work is a five layer network: 136 L 75 N 25 S 75 N 25 L, where the numbers indicate the number of nodes in the corresponding layer. L represents linear activation function, N represents tangential activation function and S represents sigmoid activation function.

4.3.1. Database Used

The quality of the unit selection voices depends to a large extent on the variability and availability of representative units. It is crucial to design a corpus that covers all speech units and most of their variations in a feasible size. The experiment is conducted on Telugu and the voice is recorded by a female speaker. All sentences are recorded in a professional studio and the sentences are read in a relaxed reading style, which is between "formal reading style" and "free talking style", at moderate speaking rate. Recordings are performed in a soundproof room with close-talking microphone. The parameters extracted from the speech signal were 25 coefficient Mel-Cepstral Coefficients (MCEPs) and f_0 using ESPS formant extraction [14] with 25 milliseconds frame size and 5 milliseconds frame shift. The speech database has been phonetically labeled using Ergodic hidden Markov models (EHMM) [15], which is well tuned to automatic labeling for building voices in Festvox [7] framework. Out of 1632 utterances, 1468 utterances were used as a part of training and the remaining utterances were used for testing.

4.3.2. Evaluation

To evaluate synthesis quality, Mel Cepstral Distortion (MCD)[16] is computed on held-out data set. The measure is defined as

$$MCD = (10/ln10) * \sqrt{2 * \sum_{i=1}^{25} (mc_i^t - mc_i^e)^2} \qquad (2)$$

where mc_i^t and mcd_i^e denote the target and the estimated mel-cepstra, respectively. MCD is calculated over all the MCEP coefficients, including the zeroth coefficient. MCD is calculated over all the MCEP coefficients, including the zeroth coefficient. Lesser the MCD value the better it is, and informally we have observed that a difference of 0.2 in MCD value produces difference in the perceptual difference in quality of synthetic speech. The MCD value we obtained is 6.53.

5. CONCLUSION

In this paper we have discussed issues involved in building multilingual screen reader; such as extraction of text from applications, converting extracted text into system readable and understandable form and building small footprint synthesizer for converting text into spoken form. The quality of the synthesis is evaluated using objective measures.

6. REFERENCES

[1] K. Krishnan, "Acharya software for the visually handicapped, iit madras," 2003.

[2] Kurzweil., "Text reader with tts," *http://www.kurzweiledu.com/support k3000win.asp*, 2004.

[3] NAB NewDelhi., "Screen access for all, (safa): Screen reader in indian languages," *http://safa.sourceforge.net.*

[4] E. V. Raghavendra, B. Yegnanarayana, A. Black, and K. Prahallad, "Building sleek synthesizers for multi-lingual screen reader," *accepted at Interspeech*, September 2008.

[5] E. V. Raghavendra, B. Yegnanarayana, A. Black, and K. Prahallad, "Speech synthesis using approximate matching of syllables," *submitted at IEEE workshop on Spoken Language Technologies.*, December 2008.

[6] E. V. Raghavendra, D. Srinivas, B. Yegnanarayana, A. Black, and K. Prahallad, "Global syllable set for speech synthesis in indian languages," *submitted at IEEE workshop on Spoken Language Technologies.*, December 2008.

[7] A. Black and K. Lenzo, "Building voices in the festival speech synthesis system," 2000, http://festvox.org/bsv/.

[8] Kishore Prahallad Anand Arokia Raj, "Identification and conversion of font-data in indian languages," *in International Conference on Universal Digital Library*, November 2007.

[9] Sathish Chandra Pammi Santhosh Yuvaraj Mohit Bansal Kishore Prahallad Alan W Black Anand Arokia Raj, Tanuja Sarkar, "Text processing for text-to-speech systems in indian languages," *in 6th ISCA Workshop on Speech Synthesis (SSW6) August 2007*, August 2007.

[10] Hunt, A.J. and Black, A.W., "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proceedings of ICASSP-96*, Atlanta, Georgia, 1996, vol. 1, pp. 373–376.

[11] A. Black, H. Zen, and K Tokuda, "Statistical parametric synthesis," in *Proceedings of ICASSP*, 2007, pp. IV–1229–IV–1232.

[12] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and Kitamura T., "Speech parameter generation algorithms for hmm-base speech synthesis," *in Proceedings of ICASSP*, 2000.

[13] C. Fatima and G. Mhania, "Towards a high quality arabic speech synthesis system based on neural networks and residual excited vocal tract model," *Signal, Image and Video Processing*, vol. 2, no. 1, pp. 73–87, January 2008.

[14] ESPS, "Esps source code from the esps/waves+ package," 2009, [Online; accessed 9-April-2009].

[15] K. Prahallad, A.W. Black, and R. Mosur, "Sub-phonetic modeling for capturing pronunciation variations for conversational speech synthesis," in *Proceedings of ICASSP*, France, 2006.

[16] Tomoki Toda, Alan W Black, and Keiichi Tokuda, "Mapping from articulatory movements to vocal tract spectrum with gaussian mixture model for articulatory speech synthesis," in *in 5th ISCA Speech Synthesis Workshop*, 2004, pp. 31–36.