Sādhanā Vol. 27, Part 1, February 2002, pp. 113–126. © Printed in India

Indian accent text-to-speech system for web browsing

ANIRUDDHA SEN and K SAMUDRAVIJAYA

School of Technology and Computer Science, Tata Institute of Fundamental Research, Homi Bhabha Road, Bombay 400 005, India
e-mail: {asen,chief}@mailhost.tifr.res.in

Abstract. Incorporation of speech and Indian scripts can greatly enhance the accessibility of web information among common people. This paper describes a 'web reader' which 'reads out' the textual contents of a selected web page in Hindi or in English with Indian accent. The content of the page is downloaded and parsed into suitable textual form. It is then passed on to an indigenously developed text-to-speech system for Hindi/Indian English, to generate spoken output. The text-to-speech conversion is performed in three stages: text analysis, to establish pronunciation, phoneme to acoustic–phonetic parameter conversion and, lastly, parameter-to-speech conversion through a production model. Different types of voices are used to read special messages. The web reader detects the hypertext links in the web pages and gives the user the option to follow the link or continue perusing the current web page. The user can exercise the option either through a keyboard or via spoken commands. Future plans include refining the web parser, improvement of naturalness of synthetic speech and improving the robustness of the speech recognition system.

Keywords. Web reader; text to speech; speech synthesis; speech recognition; Indian accent; human computer interaction.

1. Introduction

In this information age, storage and retrieval of information in a convenient manner has gained importance. Because of the near-universal adoption of the World Wide Web as a repository of information for unconstrained and wide dissemination, information is now broadly available over the internet and is accessible from remote sites.

However, the interaction between the computer and the user is largely through keyboard and screen-oriented systems. In the current Indian context, this restricts the usage to a minuscule fraction of the population, who are both computer-literate and conversant with written English.

In order to enable a wider proportion of the population to benefit from the internet, there is need to provide some additional interface with the machine. Speech, being a natural mode of communication among human beings, can also be a convenient mode of interaction with a computer. Internationally, efforts are already on to combine hypertext navigation with spoken language (Lau *et al* 1997). This can be of particular significance in our country where the rate and level of literacy are quite low. Coupled innovatively with visuals, speech and sound can

add a new dimension for conveying information to the 'common' man. Audio information can also be broadcast (e.g. at community centres) or made available on demand by telephone, thereby greatly expanding the range of end-users.

It is desirable that the human-machine interface permits communication in one's native language. This is an important issue in a multi-lingual country such as India. If the human-oriented information over the internet can be coded in a form suitable for display in a script familiar to the user and also for reading out in an acceptable language and accent, the computer can process such hypermedia document and provide the information appropriately to a large number of users.

This paper describes a web reader which can 'read out' appropriately coded HTML pages in Indian languages and accents. To start with, Hindi and 'Indian English' have been chosen. Choice of Hindi in India is automatic. English, however, is the pre-dominant internet language and will continue to be so at least in the near future. The web reader thereby caters for English also, but reads them out in an accent which is considerably more acceptable than the standard English accent to an average Indian.

2. Overview of the web reader

The function of the web reader is to download the user-specified hypermedia document, parse the document to extract textual information, feed the text to a text-to-speech system (TSS), and play the synthesized speech under user control. Figure 1 shows a functional diagram of the web reader. Its core components are an HTML document processor and a TSS. A java script downloads the web page specified by the user. This document is processed by the standard generalized markup language (SGML) parser which validates the document and provides a text representation of the elements of the document. This output is further processed by a text parser to extract information which can be fed to the TSS. The text processor module of TSS generates a sequence of phonemes and stress markers, which is fed to the speech synthesis module. The resultant speech signal is played out on a speaker or a headphone. Whenever a hypertext link is encountered in the document, the web reader reads out the title of the link. Then the user can select one of the several options for navigating through the HTML tree either using a keyboard or via spoken commands. The default behaviour of the browser is to wait for a pre-determined period of time and then continue with reading out the current document. For reading out Hindi, the text must be available in the web page, in a suitably coded form for display in the Devanagari script. Subsequent processing is similar to Indian English, except that the Devanagari script, coded in Roman alphabet, has to be interpreted. Section 6. gives some information about display/printout in Indian languages.

Details of the TSS and the HTML document processor are given in the following sections.

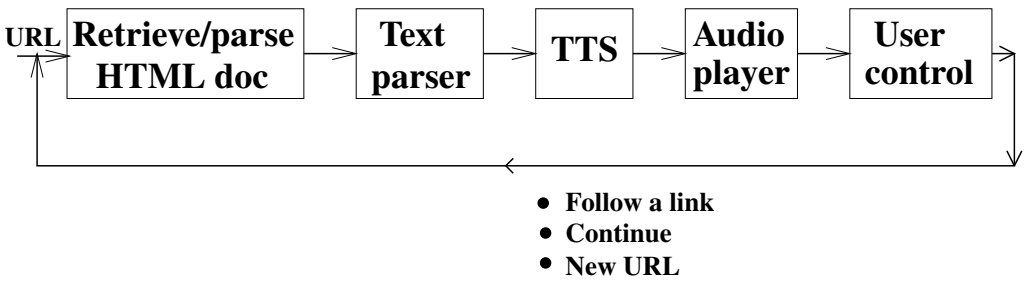

Figure 1. A block diagram of the web reader.

3. Text-to-speech system

3.1 *Background*

Text-to-speech (TSS) conversion has to be performed in two steps: (a) text to phoneme conversion and (b) phoneme to speech conversion. In the second step, we face the 'core' problem of speech synthesis, viz. that acoustic manifestations of phonemes are context dependent and transitions between phonemes carry vital perceptual cues. Synthesis of continuous speech therefore cannot be done just by 'cut-and-paste' of individual words/phonemes.

Two parallel approaches to overcome this problem are as below.

(a) *Concatenation* method uses cut-and-paste, but larger splices of natural speech, e.g. syllables, diphones are selected as units, so that immediate contextual effects are captured. Also, the splices are to be concatenated at 'steady-state' of speech.
(b) In the *Generative* method, speech is generated from some kind of speech production model and the control parameters of the model are varied usually by context-dependent rules. '*Formant*' synthesizer is a very widely used generative synthesizer which uses a speech production model based on 'formant' (i.e. resonance) frequencies.

In Indian language synthesis, both methods were used with success. A Hindi/Urdu TSS was developed at Deccan College, Pune, using demi-syllable concatenation (Bhaskararao *et al* 1994). Indian Statistical Institute, Calcutta has developed a concatenation based Bengali synthesizer and has put it into applications (Dan *et al* 1995). On the other hand, at the Tata Institute of Fundamental Research (TIFR) we have developed a synthesizer for Indian languages, using formant synthesis (Furtado & Sen 1996). The Central Electronic Engineering Research Institute, New Delhi has also developed Hindi synthesizers using the same technique (Agrawal & Stevens 1992).

Each method has its pros and cons. In general, concatenation synthesis is better '*within a segment*' whereas formant synthesis is better '*across the segments*'. Without going into a detailed comparative study, we can say from practice that for quality concatenation synthesis, we need big segment inventory – currently an unlikely proposition for Indian languages. Also, a generative method is more flexible for adaption to different languages and that is an added incentive in India, a multi-lingual country.

3.2 *Overview of TIFR TSS*

TIFR TSS is based entirely on software. It is capable of working virtually on any platform. There are versions on Unix/Linux and also on DOS/Win98. It can be adapted to any standard multi-media kit (e.g. 'Soundblaster'). The synthesizer works in 'real-time', meaning that it is possible to 'continuously read out' from a passage by multiple-buffered I/O. It currently supports four voice types and variable speaking rate.

As mentioned earlier, the text-to-speech conversion is done in two steps: (i) the front end *text analyzer* converts text into a phoneme string, (ii) speech waveform is then generated from this phoneme string. The TSS currently can optionally work for Hindi or Indian English, just by changing the front-end text analyzer which generates the phonemes and stress markers corresponding to the given Hindi/English text. The Hindi and Indian English text analyzer were built in line with Bhaskararao *et al* (1994) and Sen (2000) respectively.

The phoneme-to-speech conversion is realized in two stages: (a) Conversion of the specified phonemes into corresponding time-varying acoustic–phonetic parameters, by means of a set of

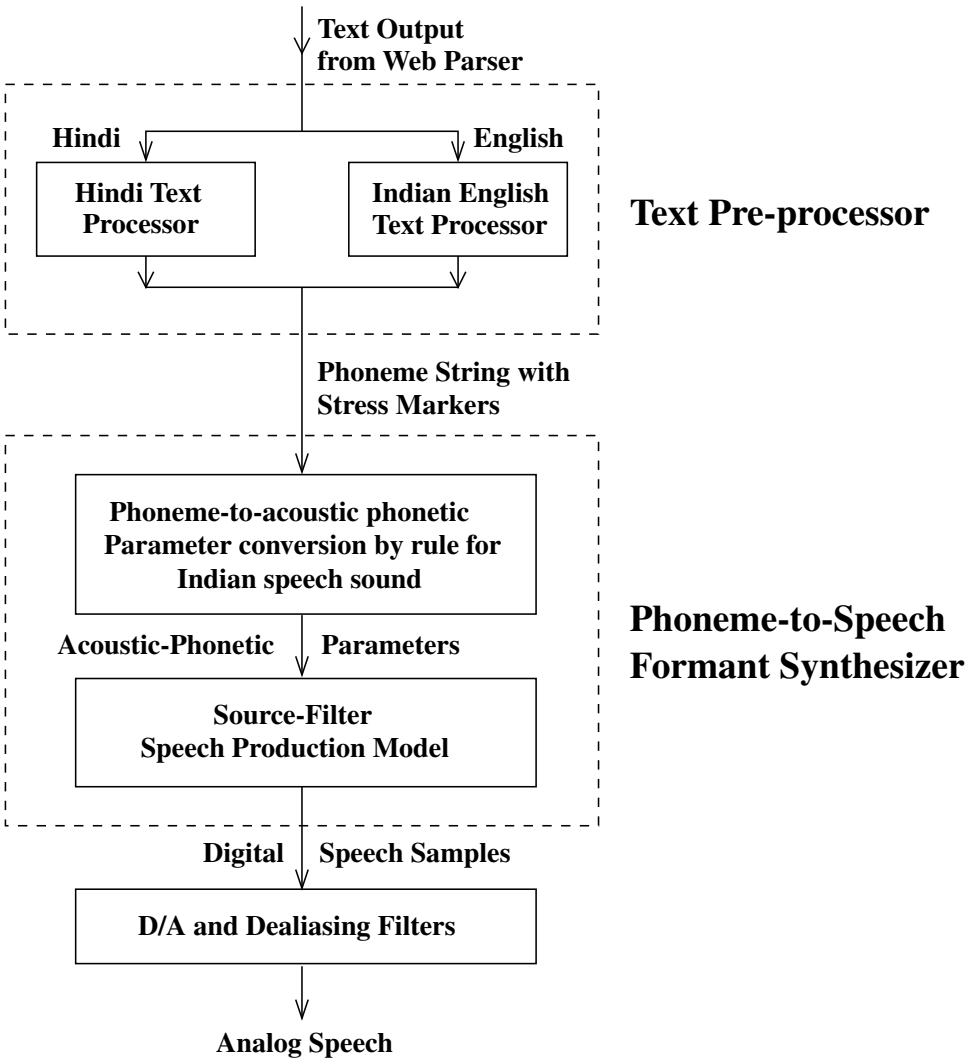

Figure 2. Text-to-speech system for the web reader.

context-dependent rules, suitable for Indian pronunciation and (b) conversion of the generated acoustic–phonetic parameters into corresponding speech (digital samples) by a source-filter speech production model. A preliminary version of this phoneme-to-speech conversion system was described by Furtado & Sen (1996). Subsequently, rules and data were continuously improved. A few strategies were also changed, which are described in § 3.4. This stage is virtually identical for Hindi and Indian English. If any additional (structurally similar) Indian language is to be incorporated into the TSS in the future, minimal changes will be needed in the phoneme-to-speech subsystem. Only the text analyzer appropriate for the new language needs to be added.

Text analyzers and the two stages of the phoneme-to-speech conversion system are discussed in the following sub-sections in some detail. Figure 2 shows the schematic of the entire synthesizer with Hindi/Indian English text analyzers as alternatives.

Figure 3. The pronunciation of the number 1312345 in a Hindi document.

3.3 *Text analyzer*

Input to this stage is Hindi/English text and the output is a set of phoneme symbols and stress markers, indicating pronunciation and accent in either Hindi or Indian English. The phoneme repertoire used is the same at the moment for Hindi and Indian English. It is in a sense a superset of the phonemes of both languages, with considerable overlap. This helps to pronounce English words used in Hindi text and Indian names embedded in English text. The stress markers currently include fullstop, semi-colon, comma, interrogation, exclamation and end-of-word symbols.

Hindi and Indian English text analyzers are two completely different units. But the underlying methodology of text processing is by and large the same for both and is being described hereafter. The special aspects of each are discussed, when appropriate.

The textual message is first passed on to the main parser, which segments the text and classifies each segment into one of the following:

(a) Date, (b) time, (c) currency, (d) alphanumerics (alphabets and numbers in the same word), (e) acronym (single characters punctuated by periods), (f) abbreviations (text terminated by a period), (g) special characters (e.g. +, -), (h) numbers (digits and optional decimal point), and (i) text words i.e., the words of the language and names. Processing the last mentioned category is the main task of the text analyzer.

After classification, a detected date, time or currency is appropriately interpreted and the pronunciation generated. An alphanumeric or acronym is pronounced character by character. Pronunciation of an abbreviation (e.g. Dr.) is found from 'Abbreviation table'. (If no match is found, it is treated as an alphanumeric.) Pronunciation of each special character is obtained from 'Character table'. Special characters for punctuation (e.g. comma, fullstop, question mark) are utilised to generate proper punctuation or accent. Some special characters have different pronunciations in text and mathematical environments (e.g. – as hyphen or minus). In future, we intend to detect the environment and generate such pronunciation accordingly.

A number string is divided into fields, and keywords corresponding to billion, million, thousand etc. are inserted. For example, 1312345 is pronounced as *One million, three hundred and twelve thousand, three hundred and forty five* in English and as shown in figure 3 in Hindi. Digits after the decimal point are pronounced character by character.

The conversions described so far are merely of a *routine* nature. The *core problem* is to obtain pronunciation of *language words*, a task which is non-trivial. We are currently accomplishing the task in the following steps.

(a) The word is looked up in a phonetic dictionary. We have prepared a Hindi phonetic dictionary of about 8000 words and an Indian English one of about 3000 words. The vocabulary of both, particularly of the latter, will be gradually enhanced. The dictionary access is done by indexing and binary search, which makes the look-up quite fast even on platforms with memory constraints, such as DOS. If the look-up succeeds, the pronunciation is obtained from the dictionary.

(b) If no match is found, a 'morphological analysis' is done to separate various components of the word i.e. prefixes, suffixes and roots. The prefixes and suffixes, however, have quite different connotations for Hindi and English. It is possible to have ambiguities in morphological analysis (e.g. cared = care+ed or = car+ed). In such cases, a few 'strong'

root alternatives are generated as per the letter context and each of them is searched for in the dictionary, in an order determined by some heuristic rules. If a match is found, the pronunciation of the root is obtained from the dictionary. The pronunciation of the prefixes and suffixes are then merged with it, following suitable contextual modifications.

(c) If there is no match even now, we select the 'best' root alternative and fall back to a set of 'Letter-to-Phoneme' rules. These rules are quite different for Hindi and English and are quite elaborate for the latter, as its script is not phonetic. Basic (default) pronunciation of letter(s) are obtained from a 'symbol table'. In Hindi, each character is assigned a pronunciation whereas the symbol table for English contains relevant letter clusters in addition. Prior to table look-up, context-dependent rules are applied on the text character string. These are fewer and simpler for Hindi. But decision on deletion of 'Schwa' (first letter of the Devanagari script), following a consonant, is complicated. In English, some important rules implemented so far are the ones regarding special pronunciation of word-final 'e', word initial X/Z and double consonants. Rules regarding pronunciation of 'c' (ch/sh/k) and 'g' (j/g) under various contexts are also incorporated. Clearly, such rules cannot be 'foolproof', but merely selects 'most probable' option.

(d) The pronunciation of various morphs of the word (obtained in whichever way) are then merged together to form the pronunciation of the whole word. The influences of prefix/suffix/root on each other's pronunciation are taken into consideration.

(e) Ultimately, 'Phonological Rules' are applied to the entire phoneme string thus obtained. These take care of contextual modification of pronunciation, e.g. of *Sandhis* (compound words) in Hindi. The sets of phonological rules are completely different for Hindi and (Indian) English.

An additional problem for an Indian English text processor is to properly 'pronounce' Indian names embedded in an English text. In fact, this itself is a strong motivation for using 'Indian English', rather than using standard English synthesizers commercially available. As spelling conventions for standard English words and Indian names are not identical, different letter-to-phoneme rules are needed to generate their pronunciation. Unfortunately, Indian names cannot be detected unambiguously. Also, no clear-cut convention to represent 'typical' Indian phonemes (e.g. retroflexed/dental, aspirated/non-aspirated stop consonants, nasal vowels) in Roman script is followed. To do the best in the given situation, we have adopted a method which generates a 'score', depending on the position of the word in the sentence (first or not), whether the word starts with a capital, whether it is there in the (domain specific) 'Name dictionary' and which prefixes/suffixes (if any) were detected. The method, although not foolproof, works reasonably well. Currently, we are making a statistical analysis of letter clusters present in Indian names and English words. For example, 'bh' or 'sr' clusters are rare in English, whereas 'ous', 'tion' etc. are uncommon in Indian names. This, coupled with grammatical validity check for noun, will improve this decision in future.

Figure 4 outlines the scheme for the text processor.

3.4 *Phoneme to acoustic–phonetic parameter conversion*

This module generates a set of acoustic–phonetic parameters, corresponding to the input phoneme string, for every small time interval (currently, 5 ms) of speech. The parameters are of two types: (a) fixed, which decides the voice type, and (b) variable, which corresponds to the changes of phonetic contexts during speech.

The variable parameters used are: (a) amplitude of voicing for vocal sounds, (b) three 'formant' (resonance) frequencies, (c) the corresponding bandwidths, (d) amplitude of 'frication',

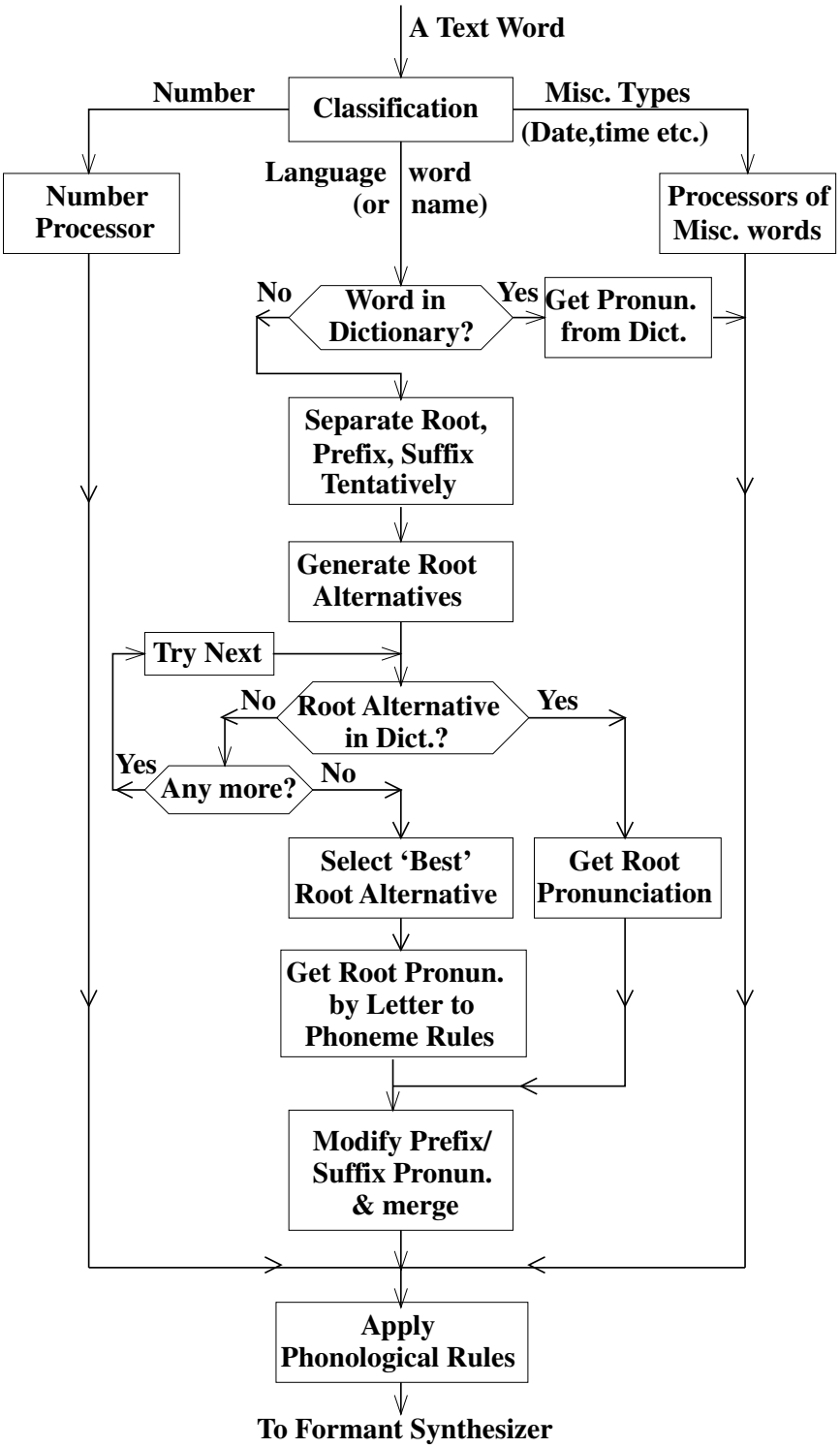

Figure 4. Text analyzer of the web reader.

for noise-like sounds, (e) five amplitudes for various bands of energy in the noise spectrum, (f) amplitude of 'aspiration', for h-type of sounds, (g) spectral tilt and 'open quotient' of voicing, to shape the spectrum, and (h) pitch.

Acoustic manifestations of various phonemes (in terms of the above parameters) are not fixed and are greatly influenced by the context. Separate sets of rules, usually corresponding to the immediate diphone contexts, were thereby formulated to generate each parameter track. To avoid 'combinatorial explosion', phonemes are grouped into different classes, according to the places of articulation (e.g. vowel, stop consonant, nasal). Rules are selected as per the two adjacent phoneme classes (e.g. silence-to-vowel, nasal-to-stop), but they operate on the phoneme data of individual elements.

The rules are encapsulated in a very compact and flexible organisation of a three-dimensional 'interpolation table'. Each entry corresponds to the current phoneme class, the next phoneme class and the parameter type. Each entry specifies an 'interpolation type' (i.e. the manner in which the parameter should be varied) in the given situation and the parameter track is computed according to the interpolation type selected. For example, the interpolation may be 'linear' (where the change is gradual) or abrupt, like a sudden jump and then a slow glide. New interpolation types are introduced and the old ones modified as per the results of our acoustic–phonetic studies.

Formant frequency is singularly the most important parameter type for speech perception and synthesis. Special care is therefore needed to generate 'formant' tracks. Till recently, we employed the following 'Modified locus equation' of Klatt (1987) to generate formant tracks:

$$F_o = F_l + k(F_v - F_l),$$

where F_o is the formant frequency at onset/offset of the vowel and F_v is the steady vowel formant frequency. F_l (called 'locus frequency') and k are constants which are determined experimentally and are tabulated for each formant of each consonant.

However, as storage is not a problem in modern computers, currently we are shifting towards storing all the relevant sets of formant tracks and then finding the desired value by table interpolation.

The main steps taken for phoneme-to-parameter conversion are:

(1) The phoneme string is parsed and each phoneme identified. Silence is appended at both ends. Each 'geminate' is replaced by a single phoneme with increased duration.
(2) Entire duration is divided into a set of 'steady-state' segments and 'transition' segments. Intrinsic duration of each segment is obtained by table look-up and is then modified as per the adjacent phoneme contexts.
    However, in order to capture the dynamics of speech better, we are currently moving away from the concept of 'steady-state'. In the new scheme, a phoneme is composed of one or more segments and transitions are 'superimposed' on each segment.
(3) For a given parameter and a given phoneme, the transition segment is first interpolated and then the steady state is interpolated. Corresponding interpolation types are obtained from the appropriate tables.
(4) Step 3 is repeated for all variable parameters for a given phoneme.
(5) Steps 3 and 4 are repeated for each phoneme.
(6) Ultimately, the pitch contour is superimposed on the utterance.

The synthesis rules employed are completely 'Indian' and all typical Indian speech sounds (e.g. retroflexed and dental stop consonants or nasalized vowels) are covered. With slight

modification of this module (and in conjunction with a suitable front-end text analyzer), synthetic speech can be generated in most Indian languages and also in Indian English. This assumes special significance in the context of accessing web information in the multi-lingual environment we envisage.

3.5 *Parameter to (digitized) speech synthesis*

This is implemented by a slightly modified version of a standard source-filter speech production model (Allen *et al* 1987; Klatt 1980). The vocal tract is modelled as a number of bandpass filters, with the 'formant' frequencies as their mean frequencies. The source of excitation to such filters is either a train of quasi-periodic pulses (for voiced sounds like vowels) or random noise (for noisy sounds like 's' or the burst of a stop consonant). For sounds like 'voiced consonants' (e.g. 'b'), both these sources are simultaneously activated.

The filters are configured in cascade to generate periodic voiced sounds and in parallel to generate noise spectrum. As aspiration noise is generated inside the glottis, it is modelled by passing random noise through *cascade* filters.

The equations of digital resonators are given by,

$$y(nT) = Ax(nT) + By(nT - T) + Cy(nT - 2T)$$

where $y(nT)$, $y(nT - T)$ and $y(nT - 2T)$ are the current and two previous output samples and $x(nT)$ is the current input sample. The resonator co-efficients A, B, C are given by:

$$C = -\exp(-2\pi B_w)T,$$
$$B = 2\exp(-\pi B_w T)\cos(2\pi FT), \text{ and}$$
$$A = 1 - B - C,$$

where F is the particular resonance (formant) frequency, B_w is the corresponding bandwidth and T is 1/(sampling rate). Currently, a sampling rate of 16,000 Hz is used.

The nasal vowels and consonants are implemented by a pole-zero pair. Their frequencies are maintained nearly the same for non-nasalized sounds and are kept suitably apart to generate the effect of nasalisation. The anti-resonator equations are given by,

$$y(nT) = A'x(nT) + B'x(nT - T) + C'x(nT - 2T),$$

where $x(nT)$, $x(nT - T)$ and $x(nT - 2T)$ are the current and the two previous input samples. Anti-resonator co-efficients A', B', C' are given by

$$A' = 1/A,\ B' = -B/A \quad \text{and} \quad C' = -C/A.$$

The frequency of the voicing source corresponds to the given pitch frequency. The shape of the source pulse is important for the "naturalness" of synthetic speech. To generate a 'natural' source, we use the standard waveform equation,

$$U_g(t) = at^2 - bt^3$$

where a, b are constants, determined experimentally.

Currently, we are conducting some experiments for better source modelling and the results have been applied successfully for generating a different voice type. This, in turn, was applied to the web reader – with different voices applied for reading different types of messages.

The other variable parameters include spectral tilt and open quotient, which are not important for intelligibility, but contributes to the quality and "naturalness" of the synthetic speech.

4. HTML document processor

The function of the HTML document processor is to retrieve the specified hypertext document from the internet and generate textual output suitable for feeding the TSS. It consists of an HTML retrieval utility, a public domain SGML parser, and a text parser developed for web reader application. A Unix utility ('wget') or, optionally, a java script is used to download the HTML document specified by the user. An SGML parser 'NSGMLS' (see http://www.jclark.com/sp/nsgmls.htm for details) validates an SGML document and prints a simple text representation of its Element Structure Information Set. A 'perl' based text parser extracts relevant information from the detailed output of NSGMLS. The text parser detects special texts such as titles, text in special font types such as bold or italic etc. As the TSS is capable of generating multiple voice types (male/female, low/high pitch), the system conveys such non-textual information to the user via specialized voice types while reading these special types of textual information. The text parser can detect some redundant information (like duplicate specification of email address) in the HTML document and discard it. Whenever a uniform resource locator (URL) is encountered within the document, the parser constructs a structure comprising the location of the URL link in the document and the URL address. It also raises a control flag at that point in the document. Such control flags are subsequently used by the web reader to provide web navigation facilities to the user as described in the next section. HTML documents sometimes contain e-mail addresses. The constituents of e-mail addresses are composed of alphanumeric characters, and are not, in general, regular words of the language. Hence the text corresponding to e-mail addresses has to be processed in a special manner for broadcast in audio mode. So, when a link to an e-mail address is encountered, the parser processes the e-mail address into separate (character) units so that the e-mail address can be spelled out by the TSS. The text parser ignores links to hypermedia documents other than text and audio (such as images, video, frame information, postscript file) so that the web contents can be delivered in audio-only output mode. The text between successive URL links is passed on in chunks of a line or a paragraph by the text parser to the TSS for synthesis of speech.

5. Web reader system implementation

The web reader is the result of integration of HTML document processor and TSS with the audio output module of the computer, and wrapping it up with control structure for web navigation. A flow chart of the human–computer interaction in a web reader is shown in figure 5.

The output of the HTML document processor is the text to be read out to the user along with control information, if any. If the output is not a URL, the text is simply passed to TSS for generating the speech signal of appropriate voice type. The normal text is read out with a default voice type. Alternate synthesized voice type is used to indicate title or emphasized text. This is a first step in the direction of indicating the type of information being conveyed to the user in audio mode. If the output of the document processor is a hyper link, the user is offered a choice either to follow the link or continue listening to the rest of the current document. At this stage, the user also has the option of returning to the document previously visited (equivalent to the back button in a browser). The system waits for 3 seconds for the user's response. If the user does not exercise his option within 3 seconds, the system follows the

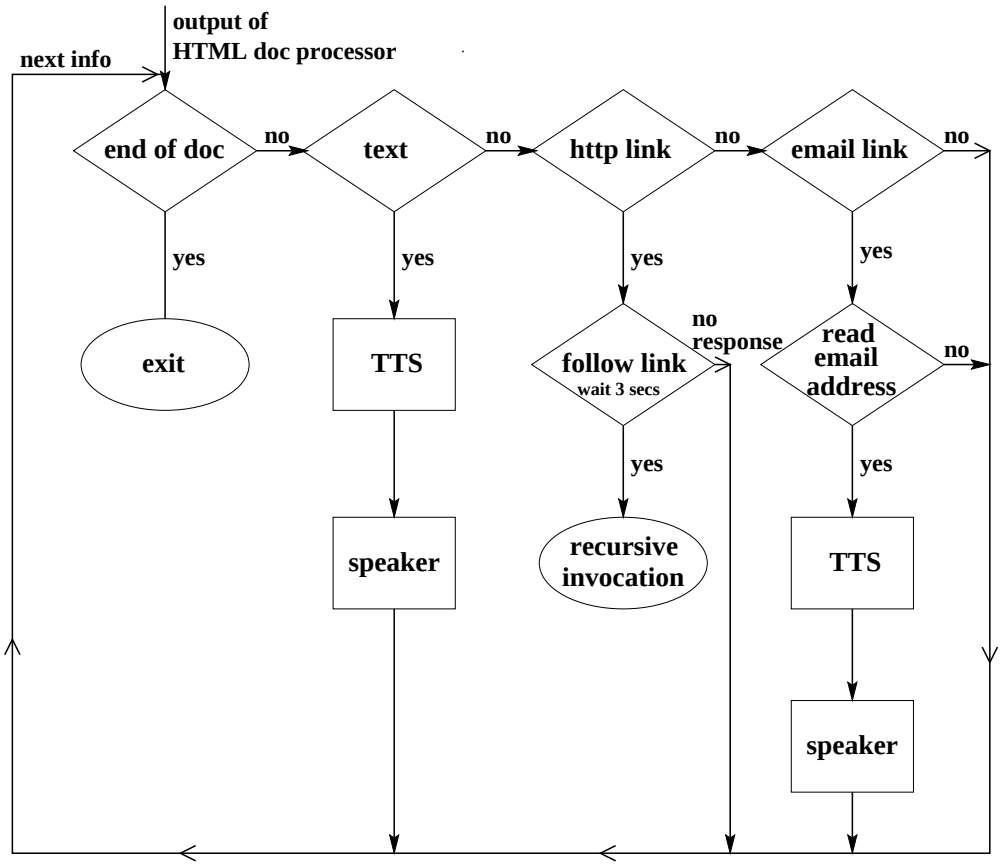

Figure 5. A flow chart of interaction of web reader with user.

default path, i.e. continues to read the current document. On the other hand, if the user chooses to follow the HTTP link, the reading of the current document is suspended, and a daughter process is spawned. The daughter process invokes a new instance of the web reader with the address of the HTTP link. When the browsing of this (new) document is over, the daughter process terminates, and the control is returned to the parent process. The parent process resumes the reading of the current document. In this manner, the user can navigate through inter-linked hyper text documents. In addition, whenever an email address is encountered in a document, the web reader offers the user a choice of listening to the email address (in spelling mode).

The process of web navigation within the web reader involves input from the user. The input is the selection of one of the 3 alternatives. The user can choose (1) to follow a hyper link, (2) to continue browsing the current document, or (3) to return to browse the previously visited web document. In case of a link to an e-mail address, the alternative option (1) is to read the e-mail address. The user's selection can be fed to the system either through a keyboard or, optionally, in voice mode, provided the system is equipped with a microphone. In the former mode, the user types in the appropriate digit to indicate his choice. In the latter mode, the user speaks the appropriate keyword, and a speech recognition module recognizes the spoken word, and passes on the choice to the control unit of the web reader. A few

isolated word speech recognition systems in Indian languages have already been developed and reported (Samudravijaya *et al* 1998; Rao *et al* 1996). The speech recognition module used with the web reader uses hidden Markov model (Rabiner 1989) to recognize the user's spoken commands.

An isolated word speech recognition system, implemented using hidden Markov models, was integrated with the web reader. This empowers the user to navigate through webpages by exercising his option, in voice mode, whenever a hyper link is encountered. The vocabulary consists of 4 words: (a) go back, (b) continue, (c) follow link, and (d) read e-mail. At any given time, the active vocabulary consists of 3 words. The words "go back" and "continue" are always active. The word "follow link" is active whenever a hyper link to another HTML page is encountered. Similarly, the word "read e-mail" is active whenever an e-mail address is encountered inside an HTML document. The user can speak the appropriate word within the active vocabulary to select the course of action.

In the current implementation, a 10 state left-to-right hidden Markov model (HMM) has been employed. During the training phase, a HMM was trained for every word using multiple repetitions of the word by 20 speakers. The public domain HTK software (http://htk.eng.cam.ac.uk/index.shtml) was used for training and testing the models. Whenever a hyperlink is encountered during web browsing using the web reader, the user is prompted (in voice mode) to exercise his option. The spoken word is matched with model of each of the 3 active words. Action is taken corresponding to the word whose model matched the spoken word best.

6. Indian language display

There are several free packages for X11 and MS-Window platforms which can be utilised for displaying and printing text in Indian language scripts. We list a few of them here.

(a) Devanagari text can be inserted within a *latex* file using *devnag* software, available at, (ftp://ftp.tex.ac.uk/tex-archive/language/devanagari/distrib/).

(b) For the same purpose, *itrans* can also be used. It is available at: (http://www.aczone.com/itrans).
Both *devnag* and *itrans* use simple transliteration scheme to represent Devanagari characters in Roman script.

(c) The GIST technology from CDAC facilitates interactive web application development in Indian languages. It follows the ISCII code to represent Indian languages in native scripts. This can be obtained from:
(http://www.cdac.org.in/html/gist/gistidx.htm).
The ISCII code also forms the basis of representing Indian language scripts in *unicode* – the universal binary code to represent the scripts of all the languages of the world. Its web address is:
(http://www.unicode.org).

(d) An ISCII plugin has been developed by IIIT, Hyderabad which enables a user to display an HTML page, coded in ISCII format, in any of the brahmi-derived scripts in any font of the user's choice. The plugin also enables a web search in Indian languages. The plugin has been tested on windows as well as Unix operating systems and with Internet Explorer and Netscape browsers. The plugin can be downloaded from
http://www.iiit.net/amba/iscii_plugin/index.html.

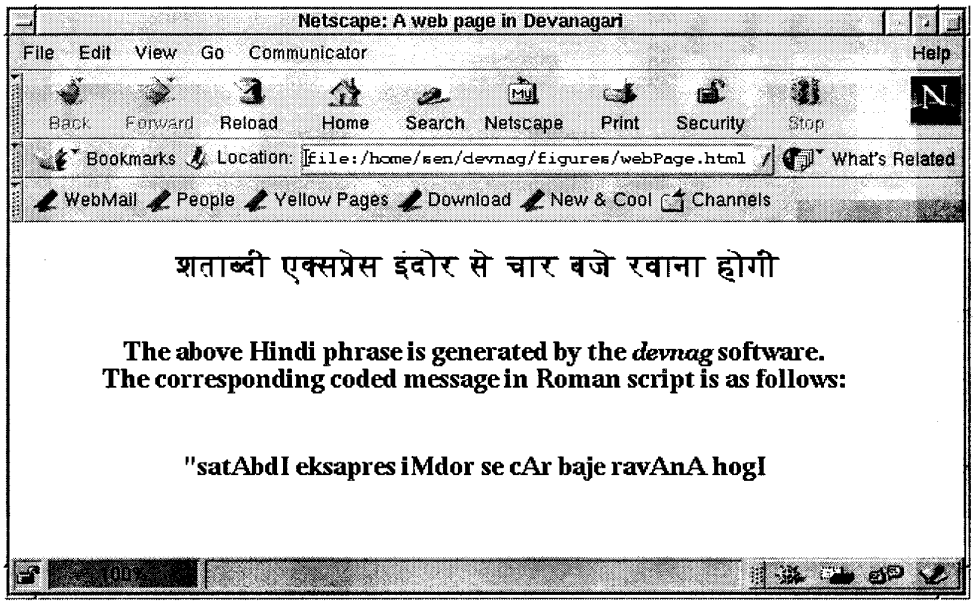

Figure 6. An illustration of Indian language display on internet.

Figure 6 illustrates the display of a Hindi web page using one such package (*devnag* software). *Devnag* represents Hindi characters using Roman script. Figure 6 shows a Hindi sentence in Devanagari script as well as the corresponding representation (in Roman script) by the Devnag software. This code in Roman script can be fed to the text analyzer to generate the sequence of phonemes and thence to synthesize speech.

Irrespective of which of the above packages is used for script display, the text analyzer can convert the text to appropriate phoneme sequence, as long as there is a one-to-one correspondence between the code and the underlying native script.

7. Conclusions

In this paper, we have presented initial efforts for utilizing spoken language as a means of communicating web information to common Indian people. The preliminary goal of integrating all the diverse elements into one system has been achieved. But this is just the beginning and there is a long way to go.

As for the individual elements, the web parser is being improved upon for better detection and classification of textual information. The web parser needs to be augmented to handle frames in web pages. The intelligibility of the TSS is good, i.e., the messages spoken are understood reasonably well. But the "naturalness" of the synthetic speech needs to be improved. It is therefore suitable for delivering only one small segment of message at a time. The quality and "naturalness" are being continuously improved up on by research in areas such as source modelling and prosody, and the results are being applied to the system.

Another issue which calls for serious attention is structuring and formatting of audio output. Ideal ways of presenting stored information in audio and video modes are not necessarily identical. Hence, there is a need to design and develop special user interfaces for delivering web information by speech.

Integrating an Indian language speech recognition system with the web reader is another task which has been undertaken. Currently, it is being used to offer users some simple choices. The menus can be expanded to ultimately offer users a keyboard-free alternative. That will help especially the blind and the semi-literate, and will open up possibilities for telephonic access of information.

We thank Prof. R K Shyamasundar for encouragement and comments. We also thank the Ministry of Information Technology, Government of India, for supporting the work reported here.

References

Agrawal S S, Stevens K 1992: Towards synthesis of Hindi consonants using KLSYN88. *Proc. Int. Conf. on Spoken Language Processing 92*, Alberta, Canada, pp 177–180

Allen J, Hunnicutt M S, Klatt D H 1987 *From text to speech: The MIT talk system* (Cambridge, MA: University Press)

Bhaskararao P, Peri V N, Udpikar V 1994 A text-to-speech system for application by visually handicapped and illiterate. *Proc. Int. Conf. on Spoken Language Processing 94*, Tokyo, Japan, pp 1239–1241

Dan T K, Datta A K, Mukherjee B 1995 Speech synthesis using signal concatenation. *J. Acoust. Soc. India*, 18: 141–145

Furtado X A, Sen A 1996 Synthesis of unlimited speech in Indian language using format-based rules. *Sādhanā* 21: 345–362

Klatt D H 1980 Software for a cascade/parallel formant synthesizer. *J. Acoust. Soc. Am.* 67: 971–995

Klatt D H 1987 Review of text-to-speech conversion for English. *J. Acoust. Soc. Am.* 82: 737–793

Lau R, Flammia G, Pao C, Zue V 1997 Webgalaxy - integrating spoken language and hypertext navigation. *Proc. Eurospeech'97*, Rhodes, Greece (European Speech Communication Assoc., France) pp 883–886

Rabiner L R 1989 A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77: 257–286

Rao P V S, Bhiksha Raj, Sen A, Mallavadhani G R 1996 A computer tutor with voice I/O in Hindi. *Knowledge based computer systems research and applications* (eds) K S R Anjaneyulu, M Sasikumar, R N Ramani (New Delhi: Narosa) pp 491–502

Samudravijaya K, Ahuja R, Bondale N, Jose T, Krishnan S, Poddar P, Rao P V S, Raveendran R 1998 A feature-based hierarchical speech recognition system for Hindi. *Sādhanā* 23: 313–340

Sen A 2000 Text-to-speech conversion in Indian English 1. *Knowledge-based computer systems 2000* (Mumbai: Allied Publishers) pp 564–575