ANUVAADHAK: A Two-way, Indian Language Speech-to-Speech Translation System for Local Travel Information Assistance

Article *in* International Journal of Engineering Science and Technology · August 2010

Source: DOAJ

CITATIONS
3

READS
3,282

6 authors, including:

Vinay Babu

2 PUBLICATIONS **4** CITATIONS

SEE PROFILE

Pradeep Kumar Narne

International Institute of Information Technology, Hyderabad

1 PUBLICATION **2** CITATIONS

SEE PROFILE

Kishore Prahallad

Apple Inc.

109 PUBLICATIONS **2,439** CITATIONS

SEE PROFILE

ANUVAADHAK: A Two-way, Indian language Speech-to-Speech Translation System for Local Travel Information Assistance

Venkata Vinay Babu Vemula, Pradeep Kumar Narne,
Mrudula Kudaravalli, Poornima Tharimela, Kishore Prahallad

International Institute of Information Technology (IIIT-H),
Hyderabad, Andhra Pradesh, India – 500032

Email: {v.v.vinaybabu, pradeepknarne, buddievergreen, taremala.poornima}@gmail.com, kishore@iiit.ac.in

Abstract:
Speech is the most natural form of human communication. In this paper, we propose Anuvaadhak, a two-way, Indian language speech-to-speech translation system, to demonstrate the possibility of designing and implementing speech-to-speech translation systems for Indian language speakers for a multitude of purposes, including but not limited to, enabling freewheeling conversations for local travel information assistance between users who cannot speak or understand each other's spoken languages! We provide a detailed description of the Anuvaadhak speech-to-speech translation system, explain its various sub-systems and measure its performance, using several evaluation parameters.

Keywords: *Speech-to-speech translation system; Indian language speech-to-speech translator; Local travel information assistance; Text-to-speech synthesis; Speech recognition; Machine translation; Phone sets.*

1. Introduction

Since time immemorial, speech has been the most widely used form of inter-species communication in the entire animal kingdom. While our human ancestors supplemented their communication needs by relying on text-based communication in the form of letters, they accorded a major importance to speech-based communication for their everyday communication needs.

On the other hand, in the present era of high-speed internet and widespread availability of low-cost, usable mobile computing resources like laptops, netbooks, tablets, mobile phones and personal digital assistants (PDAs), communication on the internet has remained mainly text-based, in the form of emails, instant messaging, social networking updates, forum discussions, blog comments, etc. The continued reliance on manual input of information and commands using keyboards and touch-screens, instead of relying on the usage of speech systems are due to the following reasons: (1) Speech systems are highly complex to build and use, limiting their usage only to users with an advanced knowledge of the field, (2) the accuracy of speech recognition is low, (3) the naturalness in the output of speech synthesizers is limited, (4) the processing power required to run the systems is high, with an accompanying high system latency to process the input and generate the output.

In this paper, we propose and use Anuvaadhak, to look at the feasibility of using speech-to-speech translators to support speech-based communication among Indian users with no common spoken (or written) language. We believe that such systems are highly necessary to support communication in today's highly-interconnected and globalized world. We were motivated to take up this challenge owing to the non-availability of multi-lingual speech-to-speech translation system for Indian language speakers, though such systems have been suggested for a few other languages [Lavie et al. (1997); Takezawa (1998); Bach et al. (2007); Waibel et al. (2003)]. These systems can be used for several domains like ticket booking, customer care centers, emergency services, etc., and languages, provided the systems are suitability "trained" with input data and optimized for the desired languages.

We describe the operation of Anuvaadhak for local travel information assistance in English and Telugu (a prominent Indian language), and calculate its efficiency and accuracy. 'Anuvaadhak' in Hindi (a major Indian language) refers to a person who translates information from one language to another. As the name suggests, our system, Anuvaadhak, has been designed for use by visitors to a city/town who want to have free-wheeling and

fluent conversations with residents and taxi drivers from this city/town in their native languages, instead of having to learn the major spoken languages of that place, either partially or fully.

The rest of the paper is organized as follows. Section 2 describes the system overview along with a typical usage scenario. In Section 3, we discuss the key features of Anuvaadhak, and later discuss its other features in Section 4. In Section 5, we show the results obtained for the three sub-systems of Anuvaadhak. In Section 6, we present the conclusion, future work and potential future uses of Anuvaadhak and other Speech-to-Speech translation systems.

2. Overview of the System

Anuvaadhak, our devised Speech-to-Speech Translation system, is composed of three parts - an Automatic Speech Recognizer (ASR), a Bi-lingual Machine Translation (MT) system and a Text-to-Speech (TTS) Synthesizer.

The design of Anuvaadhak along with its modules is presented in Fig. 1 and 2. The detailed working of various sub-systems in Anuvaadhak is described in Sections 4.1, 4.2 and 4.3.

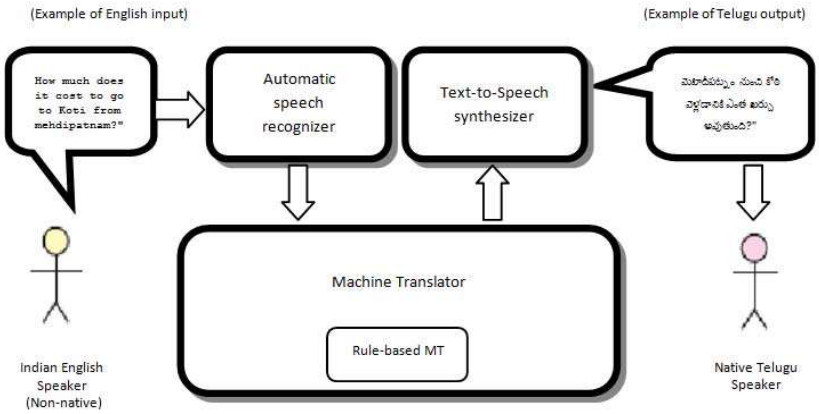

Fig. 1. Anuvaadhak: English to Telugu Speech-to-Speech translator.

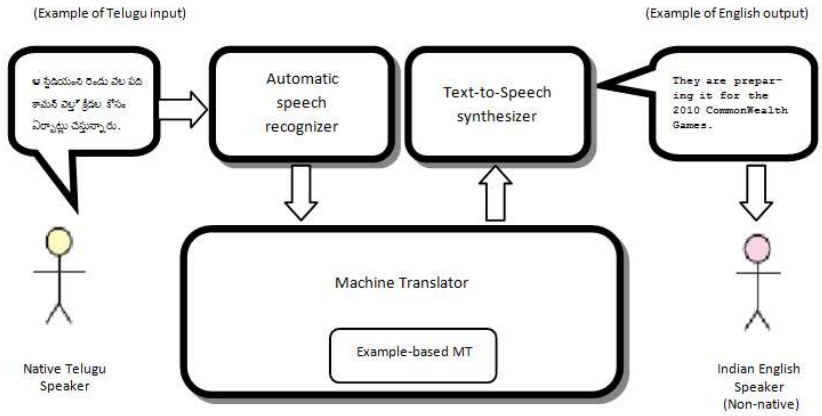

Fig. 2. Anuvaadhak: Telugu to English Speech-to-Speech translator.

2.1. *Usage scenario*

Let us consider a typical usage scenario to understand the working of our system. A native-Bengali language speaker, Mr. Prosenjit Dutta from Kolkata, India who can fluently converse only in Bengali and English, arrives at Rajiv Gandhi International Airport in Hyderabad, India by a flight to participate in a conference at Hotel Marriott in the city. A cab is sent to pick Mr. Dutta at the airport and bring him over to the venue of the conference. The driver of the cab, Mr. Anjaiah Kumar is a native of Andhra Pradesh State, India and can converse only in Telugu and Kannada.

In such a scenario, it is impossible for a fluent conversation to happen between the two speakers because they are unaware of the languages of each other. The only method of communication right now is, if one of the two parties in question 'tries' to speak in a language known to the other. We are well aware about the ensuing difficulty in communication and potential miscommunication arising in such situations.

Additionally, if Mr. Dutta wants to visit different tourist spots in the city, it becomes difficult for him to enquire and gather information about the places by speaking to different people in the city who cannot conserve in Bengali or English, due to either a lack of knowledge and/or a lack of fluency in that language.

Anuvaadhak aims to bridge this gap and provides a solution to mitigate such situations which can arise in our everyday life. It can be easily deployed on a handheld PDA or a smart phone. Since it uses a simple user interface, even users with a non-technical background can easily use this system, with little or no training. Through Anuvaadhak, we aim to provide Indian language speakers with a portable and easy-to-use Speech-to-Speech Translation system.

3. Key Features

3.1. *Generation of text through automatic speech recognition*

An Automatic Speech Recognizer (ASR) is computer-based system used to convert speech input into text output. In order to be created, a real-time speech recognizer requires the following two features: (1) representation of domain knowledge and (2) use of good search techniques. This knowledge representation should take into account all possible representations of knowledge and define methods for identifying and eliminating noise at the system input. Good searching and matching techniques ensure that the ASR is able to accurately correlate speech (spoken words) to the corresponding text.

To build the knowledge representation, a finite state representation has to be created with paths that represent valid sentences. Multiple phonetic representations on the same word have to be represented in the pronunciation dictionary. Rules on the word boundary are required to help the speech decoder detect the words correctly. This process of making the system 'learn' the sound units of our domain is referred to as training.

Prior to searching and matching, the speech has to be initially digitized and segmented into acoustic features. Consequently, the acoustic segments are matched with the available phone set to correctly identify the phone corresponding to each segment. In order for a phone to be associated with a particular segment, the former must have the maximum probability match and satisfy valid sentence rules. The matching process also referred to as decoding involves matching each spoken word, to a word from the dictionary. This is done by comparing the phonetic representation of the word - which consists of a combination of consonants (voiced and unvoiced), consonant stops, vowels and fricatives - to the list of available words, at the phone level [Lowerre (1976); Erman et al. (1980)].

3.1.1. Precautions

In order to ensure that the ASR system doesn't give erroneous outputs, we need to take the following precautions. When speaking into the microphone to give the speech input to the system, care must be exercised to ensure that the speech input is distinct, clear and free of emotions. Ideally, the training set for the ASR system should cover all possible utterances of each word in the language and be sufficiently trained for a large number of users whose speech characteristics can be considered as a representative of the speech characteristics of (nearly) all potential users of this system. Since it is difficult to completely satisfy the above criteria in real-time implementations of Speech Recognition systems, we suggest that a user of this system should avoid speaking with the context or the environment in mind, which might lead to higher values of Word Error Rate.

3.1.2. Tools

3.1.2.1 Sphinx

We use Sphinx[1], a state-of-the-art HMM-based speech recognition system developed at Carnegie Mellon University, to build Automatic Speech Recognition (ASR) modules for our domain. Sphinx consists of a set of

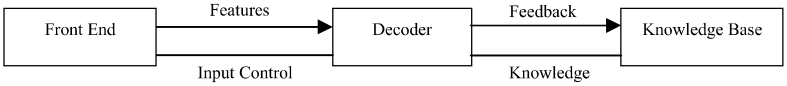

Fig. 3. Architecture of Sphinx System.

[1] Sphinx: Free tool to build Speech Recognition Systems. (Online) http://cmusphinx.sourceforge.net/

libraries written in C language, to perform various functions of speech recognition. Sphinx permits decoder reconfiguration at run-time, and allows addition of new language models and word pronunciations.

The various components of Sphinx architecture (see Fig. 3) are:

- **Feature Extractor:** A Feature Extractor extracts features from input data, enabling the decoder to read the data. It also performs Fourier analysis and noise reduction, and indicates the start and the end of the data segment for further processing.
- **Acoustic Models:** Acoustic Models deal with phonemes and changes in sound with time. Each phoneme has a sequence of states. During run-time, frames of audio are taken and compared with phoneme states to find the appropriate phonemes.
- **Language Models:** Language Models deal with probability, when there are collections of words. Internally, Sphinx uses tri-gram models that are composed of sequences of three words. Probabilities are assigned to a sequence of words, by combining bi-grams and tri-grams.
- **Pronunciation Dictionary:** Pronunciation Dictionary is composed of a list of words and a sequence of phones of the specific domain.
- **Decoder:** A Decoder possesses knowledge of word pronunciations. It takes input from the front-end, combines it with data from the knowledge base and performs a search to find the most possible sequence of words.

3.1.2.1 SphinxTrain

SphinxTrain[2] is a collection of 40 tools for acoustic model training of Sphinx 2, 3 and 4, and provides a set of Perl scripts to perform this training. The following inputs are used to train Sphinx, using SphinxTrain:

- **Phone list:** Each phone used in our domain is specified in this list of phones.
- **Dictionary:** A Dictionary provides the mapping of each word in our vocabulary with the corresponding phones.
- **Filler dictionary:** Filler dictionary consists of words like silence, breath, cough etc., which must be deleted from the speech signals, prior to processing.
- **Transcripts:** Transcripts are the text representations of each audio file used for training the ASR. They are used to match each audio file's content with words in the dictionary.
- **Speech Files:** Speech files corresponding to the transcripts are used for training the system. We use Audacity[3], a free and open-source audio recording and editing tool, to record speech in 16 bit, 16 KHz mono type files.

The trainer uses a set of sample speech signals to learn the parameters of the sound unit models called as a training database.

3.2. *Machine translation from one language to another*

A basic machine translation system makes a simple word substitution to convert text from one language to another. The translation process involves decoding the meaning of the source text and re-encoding the meaning in the target language. Word meaning refers to the concept or sense of the source text.

Machine Translation Systems can be built using several approaches like Rule-based, Statistical-based, and Example-based systems. Our system uses Rule-based translation to translate from English-to-Telugu and Example-based translation to translate from Telugu-to-English. In Rule-based translation, we perform source language text reordering, to match target language syntax, and word substitution. On the other hand, in Example-based translation, we map examples of text in source language to the corresponding representations in destination language, to achieve translation.

We use Natural Language Toolkit (NLTK)[4] to build the Machine Translation sub-system for our domain, using tree-based mapping between source language text and destination language text.

3.3. *Text-to-speech synthesis*

A Text-to-Speech (TTS) Synthesizer synthesizes speech in a particular language from the corresponding text in the same language. A good quality TTS synthesizer produces speech that sounds like human voice and can be easily understood by human users. The internal working of a TTS system is illustrated in Fig. 3.

The text which has to be synthesized into an equivalent speech output is given as the input to the TTS system. The system first normalizes this input text and then represents each word in the phonetic notation. Using this representation, the text is then divided and marked into prosodic units (phones). Since automatic label

[2] SphinxTrain: Trainer for Sphinx. (Online) http://www.speech.cs.cmu.edu/sphinx/tutorial.html
[3] Audacity: Open Source Audio recording and editing tool. (Online) http://audacity.sourceforge.net/
[4] Natural Language Toolkit. (Online) http://www.nltk.org/

generation sometimes leads to inaccurate boundary representations for phones, we use WaveSurfer[5] for manually verifying and correcting the transcription labels generated by our TTS system. Fig. 4 shows the WaveSurfer representation of an English sentence used for training our system.

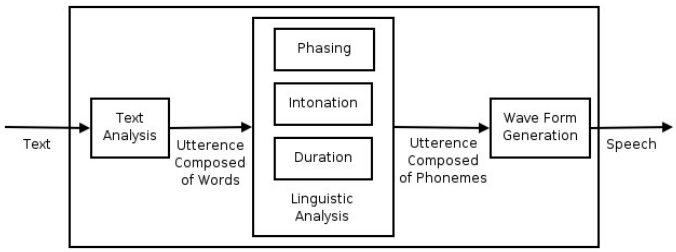

Fig. 4. Block diagram of a Text-to-Speech synthesizer.

This prosodic unit information of the input text is used to generate speech corresponding to the input text, based on the pre-existing phoneme-to-sound mapping present in the synthesizer [Dutoit (1997); Black and Lenzo (2003)]. We use Festival[6] and Festvox[7] tools to build the TTS systems for our domain.

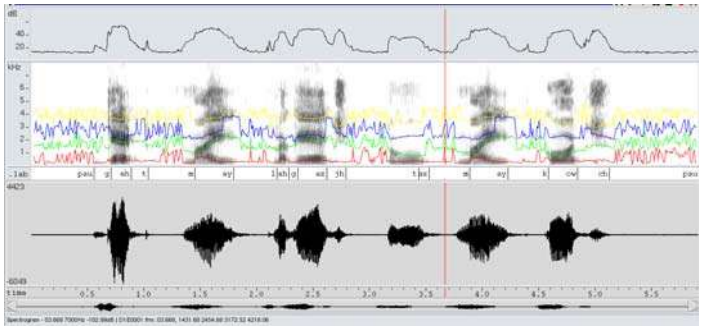

Fig. 5. WaveSurfer representation of an English sentence used to train Anuvaadhak. Here we can see the Power Plot, Spectrogram, Pitch Contour and Formant Plots of the speech file corresponding to this sentence, along with labels and label boundaries for each phone.

4. Other Features

4.1. *Phones used*

Speech Communication in any language spoken by humans can be represented as a series of basic units called phones. In order to build a system that can be used for real-time communication purposes, it is essential to extensively train our system on (nearly) all the phones in that language.

4.2.1 Telugu phone sets

The phones in Telugu language are:

"d:h"	"chh"	"nj~"	"ng~"	"r:"	"ei"	"t:h"	"gh"	"oo"	"au"	"kh"	"ph"	"t:"
"ii"	"uu"	"l:"	"h"	"rx"	"rx~"	"b"	"shh"	"bh"	"sh"	"d:"	"dh:"	"nd~"
"v"	"l"	"ai"	"t"	"aa"	"g"	"y"	"o"	"ch"	"p"	"j"	"jh"	"r"
"i"	"s"	"m"	"th"	"e"	"n"	"k"	"u"	"d"	"n:"	"a"	"a:"	"qs~"
"tra"	"t:ra"	"SIL"										

We use the following Telugu phones for Anuvaadhak:

“n”	“a”	“e”	“l”	“g”	“aa”	“k”	“l”	“t”	“s”	“u”	“k”	“r”
“oo”	“ei”	“ch”	“dh”	“r”	“m”	“d”	“ai”	“n:”	“kh”	“nd~”	“ii”	“y”
“uu”	“p”	“y”	“a:”	“tra”	“t:ra”	“SIL”	“j”	“ii”	“l:”	“h”	“d:”	“jh”
“nd~”	“l”	“dh”	“uu”	“t:”	“b”	“dh:”	“v”	“sh”				

[5] WaveSurfer: Sound visualization and manipulation tool. (Online) http://www.speech.kth.se/wavesurfer/
[6] The Festival Speech Synthesis System. (Online) http://www.cstr.ed.ac.uk/projects/festival/
[7] Festvox. (Online) http://www.festvox.org/

4.2.2 English phone sets

The phones in English Language are:

"aa"	"ae"	"ah"	"ao"	"aw"	"ax"	"axr"	"ay"	"b"	"ch"	"d"	"dh"	"dx"
"eh"	"er"	"ey"	"f"	"g"	"hh"	"hv"	"ih"	"iy"	"jh"	"k"	"l"	"m"
"n"	"ng"	"nx"	"ow"	"oy"	"p"	"r"	"s"	"sh"	"t"	"th"	"uh"	"uw"
"v"	"w"	"y"	"z"	"zh"								

We use nearly all the phones in English language for Anuvaadhak.

4.2. *User interface*

The user interface for Anuvaadhak, shown in Fig. 6, provides users with a usable and convenient way to use our speech-to-speech translation system. It is built using Python Graphical User Interface Toolkit (PyGTK)[8] on Fedora 8 [9] Linux Distribution.

Fig. 6. Graphical User Interface of Anuvaadhak, running on Fedora 8 Linux Distribution.

5. Results

5.1. *Implementation*

We designed and implemented Anuvaadhak on a notebook running Fedora 8 Linux Distribution (Kernel version 2.6.23.1-42.fc8) with GCC[10] version 4.1.2, with Intel Core 2 Duo processor with a clock speed of 2.0 GHz and 2 GB RAM. We use a standalone microphone of good quality to record the sound, and accept input for the Automatic Speech Recognition System. We used the following software package versions to build our system: Festival (version 1.96), Festvox, Sphinx2, SphinxTrain and NLTK. The user interface was designed using PyGTK.

5.2. *Analysis of automatic speech recognizer*

Twenty sentences were randomly selected from the domain and the performance of the system was tested with human speakers, whose voices were used to train the ASR system. The results are tabulated in Table 1.

The performance of the Automatic Speech Recongnizer of Anuvaadhak is tested using Word Error Rate (WER) and Word Recognition Rate (WRR), which can be calculated using Eq. 1 and 2.

$$WER = (w_d + w_i)/w_n \quad (1)$$

$$WRR = 1 - WER \quad (2)$$

[8] PyGTK: Library for building GUI's in Python. (Online) http://www.pygtk.org/
[9] Fedora Project. (Online) http://fedoraproject.org/
[10] GCC, the GNU Compiler Collection. (Online) http://gcc.gnu.org/

where, 'w_n' is the number of words in the input spoken sentence,
'w_d' is the number of words deleted (i.e. not recognized) in the text representation of the spoken sentence, and
'w_i' is the number of words inserted (i.e. incorrectly added) in the text representation of the spoken sentence.

The average word recognition rate is 0.4548 and the average word error rate is 0.5453, indicating that on an average, the Speech Recognition system of Anuvaadhak accurately recognizes 45.48% of the words spoken and fails to recognize 54.53% of the words spoken. The relatively low word recognition rate can be attributed to several factors including ambient noise. Several other factors can come into play depending on the purpose and location of usage of the ASR system, as is demonstrated in the case of Speech Recognition employed in aircraft cockpits [Englund (2004)].

Table 1. Performance study of English ASR system.

S. No.	No. of words in input sentence (w_n)	No. of words deleted (w_d)	No. of words inserted (w_i)	Word Error Rate (WER)	Word Recognition Rate (WRR)
1	10	3	3	0.600	0.400
2	11	4	3	0.636	0.364
3	6	1	2	0.500	0.500
4	12	1	1	0.167	0.833
5	19	6	10	0.842	0.158
6	24	14	8	0.917	0.083
7	16	6	2	0.500	0.500
8	14	8	5	0.929	0.071
9	20	6	5	0.550	0.450
10	10	0	0	0.000	1.000
11	6	3	3	1.000	0.000
12	6	2	1	0.500	0.500
13	7	0	0	0.000	1.000
14	4	1	2	0.750	0.250
15	12	5	6	0.917	0.083
16	12	4	3	0.583	0.417
17	8	2	3	0.625	0.375
18	6	1	1	0.333	0.667
19	7	0	0	0.000	1.000
20	9	2	3	0.556	0.444

5.3. *Analysis of Machine Translation System*

The English-to-Telugu Machine Translation System has been evaluated using METEOR[11], an evaluation metric used to assess the reliability of such systems. Table 2 provides a performance study of Anuvaadhak's English-to-Telugu Machine Translation System, using METEOR.

The average performance score of Anuvaadhak's English-to-Telugu Machine Translation System is 0.587.

5.4. *Analysis of text-to-speech synthesizer*

A perceptual evaluation study was conducted on Anuvaadhak's TTS system. We asked 5 users to rate the synthesized speech produced as output of Anuvaadhak's TTS system for 5 random sentences from our domain, on a scale on 1 to 5 (with 1 representing unclear, artificial sounding, difficult to understand speech output and 5 representing clear, natural sounding, easy to understand speech output). The results obtained on analysis are tabulated in Table 3 and Table 4.

The average perceptual rating given by users is 3.24, for both the English as well as the Telugu TTS systems.

[11] METEOR: Automatic Machine Translation Evaluation System. (Online) http://www.cs.cmu.edu/~alavie/METEOR/

Table 2. Performance study of English-to-Telugu Machine Translation system.

Input (English Sentence)	Output (Telugu Text)	Score
I want to buy some fever and cold tablets. Let's go to the closest medical store.	నేను కొన్ని జ్వరం మరియు జలుబు మాత్రలు కొనాలి మనం దగ్గరలోనున్న మందుల దుఖాణం కి వెళాలి కి అనుకుంటున్నాను	0.61
Please tell me about the history of this city.	ఈ నగరం యొక్క చరిత్ర గురించి నన్ను చెప్తారా దయచేసి	0.39
Get my luggage to my coach.	నా కోచ్ కి నా లగేజ్ తీసుకురా	0.75
Take the vehicle inside the gate.	గేటు లోపలికి వాహనము తీసుకువెల్లు	0.64
Charminar is famous for pearls.	చార్మినార్ ముథ్యల కి ప్రసిద్ది	0.38
Take me to the Bawarchi biryani restaurant as quickly as possible.	బావర్చి బిర్యాని కి నన్ను తీసుకువెల్లు	0.39
How far is Charminar from my hotel?	నా హొటల్ నుండి చార్మినార్ ఎంత దూరం	0.80
Place that bag and suitcase inside the car's dickey.	కారు దిక్కీ లో ఆ సంచి మరియు పెట్టు	0.52
I want to know more about this city.	నేను ఈ నగరం గురించి ఎక్కువ తెలుసుకోవాలి కి అనుకుంటున్నాను	0.49
Take the right direction from here.	ఇక్కడ నుండి కుడి ప్రక్కకు తీసుకువెల్లు	0.79
This note is torn. Give me another one.	ఈ నోటు చిరిగివుంది నన్ను ఇవ్వు మరి ఒకటి	0.69
Can I book your cabs by calling your customer care number?	నేను మీ కస్టమర్ కేర్ నంబర్ ఫొన్ చేసి మీ క్యాబ్ బుక్	0.67
Get me the latest public transport catalog.	నూతన పబ్లిక్ ట్రాన్సపోర్టు నన్ను తీసుకురా	0.74
What is the entry ticket rate for Salarjung museum?	మ్యూజియము కి ప్రవేశ టిక్కెట్ రుశుము ఎంత	0.54
Will you come to the desired location, if we book your cab beforehand?	మీరు కావలసిన ప్రదేశము కి వస్తర చేసుకుంటే మేము మీ క్యాబ్ ముందు బుక్	0.34
The cost of balcony tickets is thirty rupees and the price of dress circle tickets is fifty rupees.	బాల్కనీ టిక్కెట్స్ యొక్క ధర ముప్పై రూపాయలు మరియు ఫ్యామిలి సర్కిల్ టిక్కెట్స్ యొక్క యాభై రూపాయలు	0.89
What are the popular tourist attractions in your city?	మీ నగరం లో ప్రదాన పర్యాటక ప్రదేశాలు ఎంత	0.19
The traffic near Khairatabad flyover is terrible.	ఖైరతాబాద్ ఫ్లైఒవర్ దగ్గర ట్రాఫిక్ తీవ్రంగా ఉంటుంది	1.00
How do I reach the airport from the hotel?	నేను హొటల్ నుండి విమానాశ్రయం చేరుకోగలను ఎంత	0.34

Table 3. Perceptual Evaluation Study of English TTS system.

User No.	Sentence 1	Sentence 2	Sentence 3	Sentence 4	Sentence 5	Overall
User 1	4	3	3	4	3	3.4
User 2	4	3	3	4	3	3.4
User 3	4	2	2	3	3	2.8
User 4	4	3	2	4	4	3.4
User 5	4	2	2	4	4	3.2
				Overall Rating		**3.24**

Table 4. Perceptual Evaluation Study of Telugu TTS system.

User No.	Sentence 1	Sentence 2	Sentence 3	Sentence 4	Sentence 5	Overall
User 1	4	3	3	4	3	3.4
User 2	4	3	3	4	3	3.4
User 3	4	2	2	3	3	2.8
User 4	4	3	2	4	4	3.4
User 5	4	2	2	4	4	3.2
				Overall Rating		**3.24**

6. Conclusion and Future Work

In this paper, we discuss the working of Anuvaadhak, a two-way, Indian language speech-to-speech translation system which has the potential to enable fluent conversations between non-native visitors to a place with the native speakers of that place in a language, that each of them can comfortably speak and understand.

We use a typical usage scenario to explain the real-time usage of Anuvaadhak. We then discuss the working of the internal sub-systems that comprise this system, one by one. We assess the performance of the system by conducting performance evaluation studies for each sub-system and scoring the system based on its performance. The results obtained for Anuvaadhak demonstrate that our system in its present form shows areasonable accuracy in performance. In order to further improve its accuracy, we suggest a few refinements to further reduce the low error-rates in speech recognition, machine translation and text-to-speech synthesis to make it more suitable for daily use.

An overall improvement in Anuvaadhak's performance can be obtained, while making the system more robust with further enhancements to its three sub-systems. The performance of the automatic speech recognition system can be improved by increasing the word recognition rate in both noise-free as well as noisy environment. By improving the accuracy of translation as much as possible (and obtaining close to 100% accuracy in translation if possible), the machine translation system can be enhanced. The performance of the text-to-speech synthesizer can be improved by improving the naturalness of the synthesized voice, and synthesizing speech based on an analysis of the context of the speech.

We believe that speech-to-speech translation systems will further evolve to enable inter-species communication in the future, provided that the following criteria are satisfied: (1) We should possess an extensive and accurate knowledge of the vocabulary, including phone set representation, lexicon and grammar that is representative of the intra-species communication used by each species for which speech-to-speech translation systems have to be designed and implemented, (2) We should be able to create sufficiently large training sets of data, that are representative of the actual communication of the species, and (3) We should have robust software tools and accompanying hardware to support the development of such systems.

References

[1] Bach, N., Eck, M., Charoenpornsawat, P., Koehler, T., Stueker, S., Nyugen, T., Hsiao, R., Waibel, A., Schultz, T., and Black, A. (2007). *The CMU TransTac 2007 Eyes-free and Hands-free two-way speech-to-speech translation system,* IWSLT 2007, Trento, Italy

[2] Black, A. W., Lenzo K. A. (2003) *Building Synthetic Voices* (Online) http://festvox.org/bsv/

[3] Dutoit, C. (1997) *An introduction to text-to-speech synthesis,* Springer.

[4] Englund, C. (2004) *Speech recognition in the JAS 39 Gripen aircraft – adaptation to speech at different G-loads,* Masters thesis, Department of Speech, Music and Hearing, Royal Institute of Technology, Stockholm.

[5] Erman, L.D., Hayes-Roth, F., Lesser, V.R. and Reddy, D.R. (1980) *The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty,* Comput. Surv. 12 pp. 213-253.

[6] Lavie, A., Gates, D., Coccaro, N., Levin, L. (1997) *Input segmentation of spontaneous speech in JANUS: A speech-to-speech translation system,* In Dialogue Processing in Spoken Language Systems, pp. 86-99.

[7] Lowerre, B. (1976) *The HARPY Speech Recognition System*, PhD thesis, Computer Science Department, Carnegie Mellon University

[8] Takezawa, T., Morimoto, T., Sagisaka, Y., Campbell, N., Iida, H., Sugaya, F., Yokoo, A., Yamamoto, S. (1998) *A Japanese-to-English speech translation system: ATR-MATRIX,* In Proceedings of the 5th International Conference on Spoken Language Processing, pp. 2779-2782.

[9] Waibel, A., Badran, A., Black, A., Frederking, R.., Gates, D., Lavie, A., Levin, L., Lenzo, K., Mayfield Tomokiyo, L., Reichert, J., Schultz, T., Wallace, D., Woszczyna, M., and Zhang, J. (2003) *Speechalator: two-way speech-to-speech translation on a consumer PDA*, Eurospeech 2003, Geneva, Switzerland.