Volume 3, Issue 10, October 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Voice Enabled Telephony Commands Using Gujarati Speech Recognition

Jigisha K. Patel*
Department of Computer Science,
Sardar Patel University
Gujarat, India

Pritesh N. Patel
Institute of Science and Technology for Advanced Studies and Research (ISTAR), Gujarat, India

Paresh V. Virparia
Department of Computer Science,
Sardar Patel University
Gujarat, India

Abstract-- Speech recognition deals with identifying the spoken words and convert it into equivalent text form. Many hand-held devices support voice commands for operating the devices but many few of them supports the facility in local languages. In this paper, we would like to discuss the system that supports the use of Gujarati language for telephony. We have implemented and tested a system for dialling a call to some number or person. The system is speaker independent and it has been tested for 29 words. We have used the HMM based speech recognizer Sphinx4 toolkit for the same.

Keywords-- Speech recognition, Phoneme, Hidden Markov Model (HMM), Java grammar, Lexeme

I. INTRODUCTION

Speech recognition is the area of natural language processing that identifies the spoken speech and convert it into text that can be further translated in form of a command or an action. Over the last decade, speech recognition systems are increasingly used in automated systems with spoken language interfaces. Systems with spoken language interfaces make a significant contribution to the realization of local language interfaces in the Indian scenario. There has been a substantial progress in speech technology, with today's state-of-the-art systems being able to transcribe unrestricted broadcast of speech data with good accuracy. However, the number of speech recognition systems that supports Indian languages is very small and especially for Gujarati. Nowadays, many hand-held devices like mobile phones, notebooks and tablets provides the facility of speech recognition for various operations. But few of them provides the support for local languages. In this paper, we have designed and implemented a system that can identify the commands to phone a person. The command is to be given in Gujarati language which consist of digits from zero (SaUnya) to nine (nava), and instructions like “faona krao”, “faona lagaavaao”, “Dayala krao”, “kaola lagaavaao” etc. which means “dial a phone to”. The system has been trained and tested with 29 words including the above mentioned digits and commands as well as names of 10 persons. We have used Sphinx4 toolkit.

II. INTRODUCTION TO SPHINX4

CMUSphinx toolkit is a leading speech recognition toolkit with various tools used to build speech applications. CMUSphinx toolkit has a number of packages for different tasks and applications [2]. Sphinx-4 is one of the packages provided by CMUSphinx, which is a state-of-the-art speech recognition system written entirely in the Java programming language. It was created via a joint collaboration between the Sphinx group at Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP), with contributions from the University of California at Santa Cruz (UCSC) and the Massachusetts Institute of Technology (MIT). [6] Sphinx-4 is an HMM-based speech recognizer. HMM stands for Hidden Markov Models, which is a type of statistical model. In HMM-based speech recognizers, each unit of sound (usually called a phoneme) is represented by a statistical model that stands for the distribution of all the evidence (data) for that phoneme. This is called the acoustic model for that phoneme [9]. The Figure 1 shows the architecture diagram of Sphinx 4 tool kit. When the recognizer starts up, it constructs the front end (which generates features from speech), the decoder, and the linguist (which generates the search graph) according to the configuration specified by the user. These components will in turn construct their own subcomponents. For example, the linguist will construct the acoustic model, the dictionary, and the language model. It will use the knowledge from these three components to construct a search graph that is appropriate for the task. [9] Most of these components represents interfaces. There can be different implementations of these interfaces. For example, there are two different implementations of the search manager. Then, how does the system know which implementation to use? It is specified by the user via the configuration file, an XML-based file that is loaded by the configuration manager. In this configuration file, the user can also specify the properties of the implementations. [9] Sphinx-4 currently implements a token-passing algorithm. Each time the search arrives at the next state in the graph, a token is created. A token points to the previous token, as well as the next state. The active list keeps track of all the current active paths through the search graph by storing the last token of each path. A token has the score of the path at that particular point in the search. To perform pruning, we simply prune the tokens in the active list. [9]

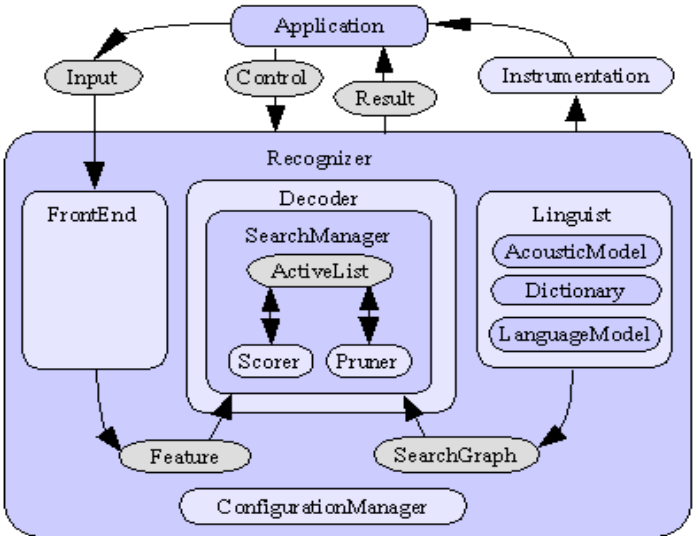

Figure 1: Block diagram representing the architecture of Sphinx4 [9]

When the application asks the recognizer to perform recognition, the search manager will ask the scorer to score each token in the active list against the next feature vector obtained from the front end. This gives a new score for each of the active paths. The pruner will then prune the tokens (i.e., active paths) using certain heuristics. Each surviving paths will then be expanded to the next states, where a new token will be created for each next state. The process repeats itself until no more feature vectors can be obtained from the front end for scoring. This usually means that there is no more input speech data. At that point, we look at all paths that have reached the final exit state, and return the highest scoring path as the result to the application. [9]

III. INTRODUCTION TO MODEL USED

Figure 2 shows the model of our system.

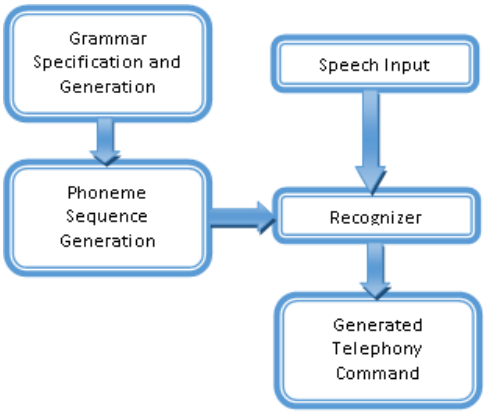

Figure 2: The model of system

Before starting the recognition, we require to specify the grammar and phoneme sequences for words. The recognizer accepts the speech input and using the knowledge of dictionary and grammar, it generates the telephony command. More details are given in implementation details [section 5].

IV. THE SYSTEM

The system is speaker independent and it can be used by the mobile operating systems to allow user to give voice command to mobile for dialing a call to person or some number. The system can also accept some variations in the pronunciation of some words in Gujarati. The system can identify any formats of command listed in Table 1 and it can be converted to the format supported by the mobile operating system. Each of the command should be augmented by "start" and "stop" word so that the mobile operating system can easily identify the beginning and ending of a command.

TABLE 1: LIST OF DIFFERENT COMMAND FORMATS

Command
Dial Ami or Dial 123
Call Ami or Call 123
Phone Moksh or Phone 1234567
Phone karo 1234 or Phone lagavo 1234

Phone karo Meeta or Phone lagavo Meeta
123 Call
123 Dial
123 Phone

V. IMPLEMENTATION DETAILS

First of all we make a grammar file which describes the words to be recognized with the command structure. The grammar file for telephony commands is as follows:

```
<DialCommand> = (<begin> <action> <person> <end>)
        | (<begin> <person> <action> <end>)
        | (<begin> <action> <digit>+ <end>)
        | (<begin> <digit>+ <action> <end>);

<begin> = (start);
<end> = (stop);

<action> = (call | dial | phone) [( karo | lagavo)];

<person> = (moksh | jigisha | ami | home | shiv | raam | raina | gokul | meeta | kishan | kirit);

<digit> = (ek | be | tran | char | panch | chha | saat | aath | nav | shunya | zero);
```

The grammar describes that the command given by user can be any of the format specified in the tag "DialCommand" where the user can speak the number first and then "dial" or "call" or "phone" and vice versa. The user can also speak the name of the contact person stored in the mobile. As the application is developed to support the use of Gujarati language in mobile, we have used the Gujarati digit for the phone number, meaning that the user has to speak number in Gujarati only. The "digit" tag in grammar file identifies any digit spoken in Gujarati.

The next step is to convert the given word to its corresponding phonemes. For this we have studied the tools like IMtool [8] which builds a consistent set of lexical and language modeling files for Sphinx (and compatible) decoders. After studying the tool and its output, we have formed our own set of phoneme sequences for each word so that it can support the Indian accent for dial command as well as the Gujarati accent for digit especially. After configuring the grammar and dictionary now the recognizer is ready to identify any possible command structure for telephony.

VI. RESULTS & DISCUSSIONS

The application have been tested by 20 speakers in the laboratory environment using the head-mounted microphone. Out of 20 speakers, 10 were females and 10 were males having age between 20 and 30. The speaker was allowed to pronounce the word however he/she likes. During the testing we found that there are some words that misrecognized as other words as shown in the Table 2. The misrecognition of such words was affecting the average recognition rate of the system. We have studied such words and added some other phoneme sequences so that the system can identify the correct word with better accuracy.

TABLE 2: WORDS MISRECOGNIZED AS OTHER WORDS (IN %)

Original Word	**Confused with word (%)**		
AATH	EK [55]	-	-
DIAL	BE [75]	-	-
KARO	GOKUL [10]	-	-
KIRIT	EK [30]	-	-
NAV	DIAL [35]	-	-
STOP	DIAL [50]	-	-
TRAN	DIAL [35]	-	-
CALL	GOKUL [30]	LAGAVO [30]	-

GOKUL	AATH [15]	CALL [50]	-
PANCH	SAAT [25]	AATH [25]	-
SHIV	SHUNYA [15]	ZERO [35]	-
CHAR	DIAL [15]	SAAT [60]	TRAN [15]
RAAM	HOME [20]	RAINA [15]	TRAN [15]
START	SAAT [15]	DIAL [15]	EK [15]

After doing the required modifications, we have analyzed the results for various aspects. One of them is based on the gender of the speaker. The Figure 3 and Figure 4 shows the accuracy of the system for individual female speakers and individual male speakers respectively.

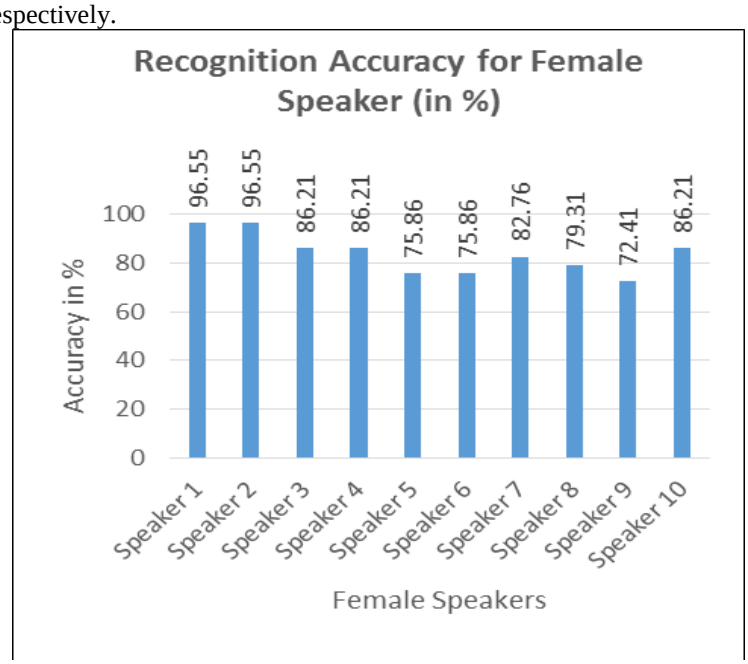

Figure 3: Recognition Accuracy for individual female speakers in %

The average accuracy for all female speakers is 83.79%.

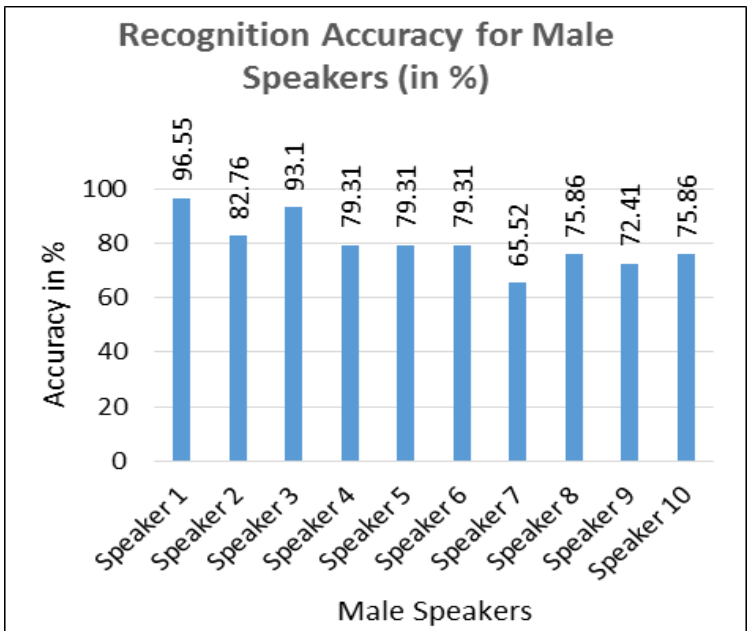

Figure 4: Recognition Accuracy for individual male speakers in %

The average accuracy for male speakers is 80%. The Figure 5 shows the accuracy of the system for all the speakers.

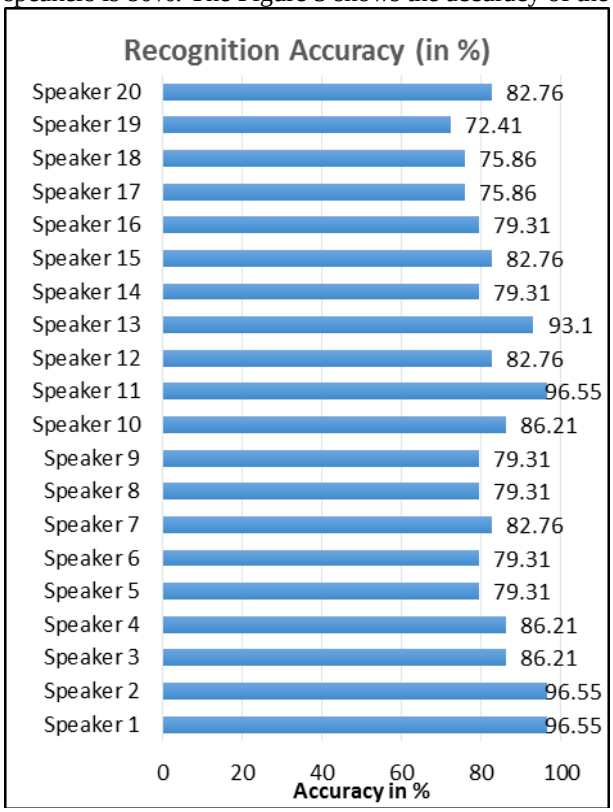

Figure 5: Recognition Accuracy for all speakers in %

As shown in the Figure 5, the minimum accuracy of the system for all the speakers is 72.41 and the highest accuracy gained is 96.55. The system gives the average accuracy of 83.62 for all the speakers. We have also calculated the recognition accuracy rate of individual words which is shown in Figure 6. The average accuracy rate of all the words is 83.62%.

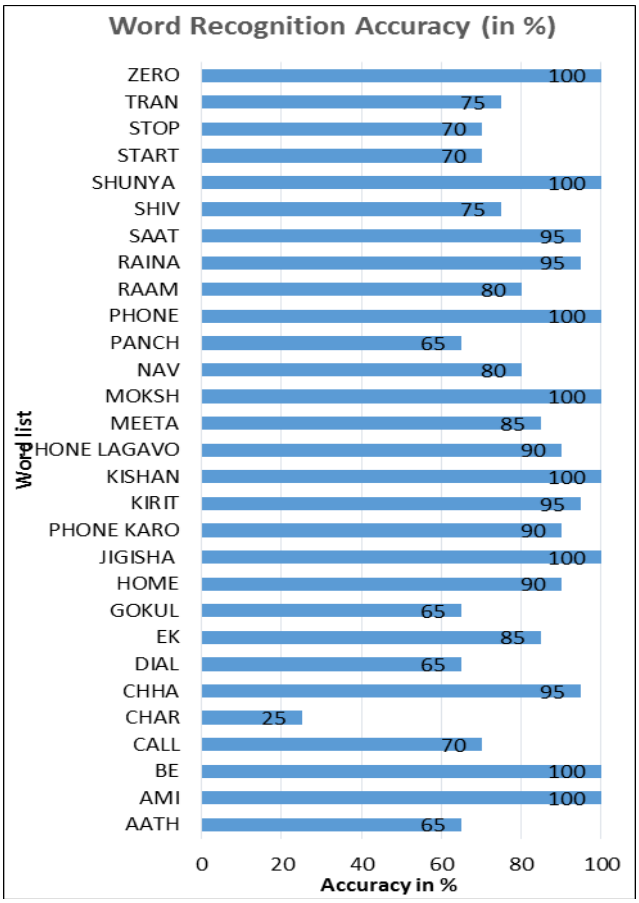

Figure 6: Recognition Accuracy for individual word in %

For the readers knowledge we have provided as shown in Table 3, the list of Gujarati lexemes used in grammar and its corresponding meaning.

TABLE 3: LEXEME LIST IN GUJARATI WITH MEANING

Lexeme	**In Gujarati**	**Meaning**
AATH	AaZ	Digit 8
AMI	AmaI	Contact name
BE	bao	Digit 2
CHAR	caar	Digit 4
CHHA	C	Digit 6
EK	Aok	Digit 1
GOKUL	gaaokula	Contact name
JIGISHA	PgaISaa	Contact name
PHONE KARO	Faona krao	Dial a call
KIRIT	ikrIT	Contact name
KISHAN	ikSana	Contact name
PHONE LAGAVO	Faona lagaavaao	Dial a call
MEETA	maItaa	Contact name
MOKSH	maaoxa	Contact name
NAV	nava	Digit 9
PANCH	paaMca	Digit 5
RAAM	rama	Contact name
RAINA	rOnaa	Contact name
SAAT	saata	Digit 7
SHIV	iSava	Contact name
SHUNYA	Saunya	Digit 0
TRAN	~aNa	Digit 3

There are many words used in our application that can have different pronunciation by different speakers. For example, "PgaISaa" can be pronounced as "PgaIsaa", "~aNa" as "~ana", "iSava" as "isava" and "C" as "Cao". The system is able to handle some of these variations in the pronunciations by different speakers. As the whole experiment has been carried out in laboratory environment, the performance may slightly decline in noisy surrounding.

VII. CONCLUSION

In this paper, we have described the system that is able to accept telephony command in Gujarati. The system is speaker independent and can be adopted by the mobile operating systems to convert the output of the system in command format acceptable by the mobile. The testing of the system has been carried out in a laboratory environment by 20 speakers comprising 10 females and 10 males. The system is giving accuracy of 83.62%. The system is also able to recognize different pronunciation of some words by different speakers. The work regarding the noise filtering and accuracy improvement is still underway.

REFERENCES

[1] Jurafsky, Martin – "Speech and Language Processing", Pearson, 2000
[2] Gunnar Fant - "Speech Acoustics and Phonetics: Selected Writings" [e-book], Springer, 2006
[3] Hinrich Schtze - "Foundations of statistical natural language processing" [e-book], MIT Press,1999
[4] Ms. Jigisha Patel, Mr. Pritesh N. Patel, Dr. P. V. Virparia – "Acoustic and Phonetic Confusions in Accented Gujarati Speech Recognition" : National Journal Of Engineering Science And Management (ISSN 2249 -0264) Bhopal
[5] Ms. Jigisha Patel, Mr. Pritesh N. Patel - "Dialectical issues in speech recognition for Gujarati language", in the Proceedings of the National Conference on Advances in Computing-2011, North Maharashtra University
[6] http://cmusphinx.sourceforge.net/sphinx4/#what_is_sphinx4
[7] http://cmusphinx.sourceforge.net/wiki/tutorialoverview
[8] http://www.speech.cs.cmu.edu/tools/lmtool-new.html
[9] http://cmusphinx.sourceforge.net/sphinx4/#architecture_and_api1
[10] http://cmusphinx.sourceforge.net/sphinx4/#download_and_install
[11] https://javacc.java.net/doc/javaccgrm.html
[12] http://cmusphinx.sourceforge.net/wiki/sphinx4:jsgfsupport
[13] http://cmusphinx.sourceforge.net/sphinx4/javadoc/edu/cmu/sphinx/jsgf/JSGFGrammar.html

[14] http://www.w3.org/TR/speech-grammar/
[15] http://docs.oracle.com/cd/E17802_01/products/products/java-media/speech/forDevelopers/jsapi-doc/javax/speech/recognition/RuleGrammar.html
[16] http://www.ling.helsinki.fi/kit/2004s/ctl310gen/L7-Speech/JSAPI/Recognition.html

