SPEECH ENABLING SYSTEM

(REGIONAL LANGUAGE CONVERSION)

A PROJECT WORK DONE AT

Rishinet Pvt.,Ltd, Bangalore-71

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

OF BHARATHIAR UNIVERSITY, COIMBATORE

SUBMITTED BY

K.SHANTHAKUMAR
Reg.No. 9838M0520

Under the Guidance of

External Guide
Mr.R.SureshRamachandran ,
Project Manager,
Rishinet Pvt.,Ltd,
Bangalore-71

Internal Guide
Dr. S.Thangasamy,Ph.D.,
Head of the Department,
Dept of Computer Science & Engg,
Kumaraguru College of Technology,
Coimbatore-641006

P-563

Department of Computer Science & Engineering
Kumaraguru College of Technology
Coimbatore,Tamil Nadu

CERTIFICATE

CERTIFICATE

This is to certify that the project work entitled "Speech Enabling System" submitted by Shantha Kumar K. Roll No.9838M0520, Student of the final year MCA (Master of Computer Applications) Kumaraguru college of Technology (Affiliated to Bharathiar University) Coimbatore is a bonafide work carried out by him under our guidance, from December 2000 to March 2001 at Rishinet Pvt., Ltd, Bangalore-560 071. This is required for the fulfillment of the degree of Master of Computer Applications of Bharathiar University for the year 2000-2001.The student will not be allowed to take the source code outside the organization.

During this period we found that, he to be a good learner and hard working. We wish him the best in all his future endeavors.

Mr. H R K Ravi
Managing Director.

Rishinet Pvt. Ltd.
302, Rishinet House, 7th Cross, Domlur Layout, Bangalore - 560 071
Tel: [91] [80] 5309799, 5309899, 5540804 Telefax: [91] [80] 5309799
E-mail: info@rishinetindia.com Website: www.rishinetindia.com

Department of Computer Science and Engineering
Kumaraguru College of Technology
(Affiliated to the Bharathiar University)
Coimbatore-641 006

CERTIFICATE

This is to certify that the project work entitled

SPEECH ENABLING SYSTEM

Done by

K.SHANTHAKUMAR
Reg.No :9838MO520

Submitted in partial fulfillment of the requirements for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** of **Bharathiar University**

S.J 8/5/01
Professor and Head

S.J 8/5/01
Internal Guide

Submitted to University Examination held on 11-05-2001

S.J 11/5/01
Internal Examiner

External Examiner

DECLARATION

DECLARATION

I here by declare that the project work entitled

"Speech Enabling System"

done at

Rishinet Pvt Ltd
Bangalore

submitted in partial fulfillment of the requirements for the award of Degree of
Master of Computer Applications
is a report of original work done by me during the period of study in

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University)
Coimbatore-641 006

Under the Supervision of **Mr. R. SureshRamachandran,** Project Manager, Rishinet Pvt Ltd, Bangalore and **Dr. S.Thangasamy, Ph.D.,** Head of the Department, Kumaraguru College of Technology, Coimbatore.

Name of the Candidate	Register Number	Signature
K.Shanthakumar	**9838M0520**	Shanth

Place : Coimbatore
Date : 8/5/2001

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I express my sincere thanks to ***Hrk. Ravi,*** *CEO & M.D, Rishinet Pvt Ltd , Bangalore for providing me the opportunity to do the project in this sophisticated establishment.*

I express my profound gratitude and thanks to my external guide ***Mr.R.SureshRamachandran,*** *Project Manager, Rishinet Pvt., Ltd for all the blessings and guidance.*

I am bound to express my gratitude to ***Dr.K.Padmanaban,*** *Principal, Kumaraguru College of Technology (Affiliated to Bharathiar University), Coimbatore, TamilNadu for his constant encouragement throughout my course.*

I wish to express my grateful thanks to my internal Guide ***Prof. S.Thangaswamy,*** *Head, Department of Computer Science & Engineering,, Kumaraguru College of Technology (Affiliated to Bharathiar University), Coimbatore, TamilNadu for constantly encouraging me to pursue new goals and ideas and had given his tremendous guidance and suggestions throughout my project.*

My Special thanks to ***Mr. CT.Venkatesh, Mr. Vikram A Bhavan*** *and* ***Ms. Shilaja,*** *for their excellent guidance, encouragement and valuable time spend during the project period.*

I am proud of my ***Family*** *for encouraging me whenever I was depressed, to face the challenges in the file and made the MCA degree possible.*

I convey my thanks to all the ***Friends,*** *and others those who helped me directly or indirectly to complete my carrier.*

K. SHANTHAKUMAR

SYNOPSIS

SYNOPSIS

The project work entitled "Speech Enabling System" for Rishinet Pvt Ltd, Banglore has been carried out to implement regional language conversion in speech technology.

Rishinet Pvt Ltd is primarily involved in research and development of "Speech Technology". This Technology is the ability of a computer to "understand" and interpret spoken words. With the recent advances in both software and hardware, it is offering an efficient and affordable alternative to traditional input devices. Researchers are also interested in *natural language processing* techniques as an extension of the speech recognition, providing a more natural and intuitive interface. The accuracy of SR software has reached well over 90%, but doesn't throw your keyboard away yet. An average of ten mistakenly recognized words on a total of hundred words still makes it far from perfect.

There are hundreds of languages in the world, and thousands of dialects but recording all of this speech-data is a very time consuming

process, and is part of the reason that speech recognizers today are only available for the major dialects of the most common languages. Today I am created a new speech recognizer for one of this regional language (Tamil) with specific Phonemes.

This Project will give a brief overview of the technology is to Understand the concept & implementation methodologies relevant to Localization of Speech Enabling System.

CONTENTS

CONTENTS

INTRODUCTION

Introduction

Innovation is nothing but a series of gradual improvement

- Japanese proverb

Speech recognition is the process by which a computer (or other type of machine) identifies spoken words. Basically, it means talking to your computer, and having it correctly recognize what you are saying.

Industry Talk:

- Gartner Group predicts that by 2002, speech recognition and visual browsing capabilities will be integrated into mainstream operating systems.
- According to a recent survey of more than a thousand chief executives in health care organizations by Deloitte & Touché Consulting, 40% planned to use speech recognition within two years.
- Microsoft invested $45 million in Lernout and Hauspie in 1997 to accelerate the growth of speech recognition in Microsoft products
- Both IBM/Lotus and Corel are delivering to the market application suites that feature speech recognition.

However, there are other forms of speech recognition based on different grammars. These grammars represent short-run solutions that can be used by more general audiences.

Scope

The scope of the document is to Understand the concept & implementation methodologies relevant to Localization of Speech Enabling Systems

- General Speech Recognition Theories
- Engines & Analysis
- SAPI
- Methodology

Project Overview

1.1 Project Overview

Objective

The Objective of the company is to develop localized version of linguistic software component which will be SAPI 5.0 Compliant.

Our work was to assist in the development of Phoneme structures for Tamil

Interactive System

Speech recognition offers certain users the best way to interact with a computer and promises to be the dominant form of human-computer interaction in the near future. Recent advances in software speech recognition engines and hardware performance are accelerating the development and acceptance of the technology. Most people are familiar with speech recognition applications based on dictation grammars, also known as continuous speech recognition. These applications require a large commitment from the user, who has to spend time training the computer and learning to speak in a consistent manner to assure a high degree of accuracy. This is too much of a commitment for the average user, who just wants to sit down and start using a product. Users of this technology tend to be those who must use it or are highly motivated to get it working for some other

reason, like people with various physical disabilities. However, there are other forms of speech recognition based on different grammars. These grammars represent short-run solutions that can be used by more general audiences.

1.2 Organization Profile

Rishinet Pvt Ltd. is an ISO9001 ,entity whose prime focus is on providing quality e-Solutions swiftly. The company commenced operations in 1996 with a team of inspired professionals who brought with them, a wealth of IT related experience and a single unified objective: to make the company a leading solution provider, with a qualitative difference in the web space.

Rishinet houses on a sprawling complex that houses a carpet area of 6,000 sq.feet with the capacity to seat 100 software professionals is not what we view as our only infrastructure.

The development facility stationed at Bangalore offers facilities such as videoconferencing, state-of-the-art multi-platform servers and connectivity that ensures swift working paces with a 24 hour global network service.

To effectively manage the overseas clientele and provide localized support, Rishinet has set up technical support and marketing offices in USA and UK.

Our corporate culture is built on fundamentally simple principles with a focus on building our team of quality people with professional values.

Business Lines

A diverse yet highly evolved expertise base is what makes Rishinet the dynamic entity that it is. Offerings have been demarcated into four broad areas that include Software Services. Co-Product Development, Professional Services and Smart Souring.

Software Services:

These services largely comprise of Consulting Services Architecture, Design, Development & Enhancement Services, Web Enabling Services, Porting & Re-Engineering Services; Sustenance Services and Testing Services.

Co-Product Development:

Rishinet is in a position to offer a range of product development services that call upon our versatile mix of skills.

The product development stream enables them to offer the following services:

- Detailing of specifications
- Conducting a technical feasibility study.
- Co-architecting, engaging in design and development.
- Planning and implementing the feature enhancement of products.
- Building in localization of products.

Professional Services:

In an industry where technical competence takes precedence over all else, we bring to every solution and optimal mix of manpower expertise and technology. Rishinet

has put in place a stringent process of selecting competent software professionals who are assigned to projects according to their skill sets and core competencies.

Smart Sourcing:

- Rishinet assists diverse international client base in offshore outsourcing of technically skilled manpower across a varied skill set base and domain industry knowledge.

SYSTEM STUDY

&

ANALYSIS

2. System Study and Analysis

2.1 Existing System

The entire process involves, recognize users voice and converting them into words. This process already available in the market, which performs only some specific languages. No other functionality or analysis is done on the word is recognized.

2.2 Limitations of Existing system

- The existing speech recognition engines have no ability to process local language.
- It is not flexible to make further study or analysis of the data and is not user friendly. Hence it takes longer time and larger effort.
- Each new user must re-train the system to reduce confusion and improve performance
- They require the greatest degree of computing resources
- The existing system does not have the advanced feature like Grammer Compiler, Voice Command, Talk Back, i.e

2.3 General theory

Human speech recognition, the mapping of the acoustic signal to a string of words, is a complex process. Many different levels of processing are involved, acoustic, sublexical, lexical, syntactic, and so on. In addition, the acoustic manifestations of linguistic units (phonemes, words) overlap in time and are greatly affected by factors such as linguistic environment, acoustic conditions, speaking style and rate, speaker identity, etc.

2.4 Engines and Analysis

Didfferent Types of Speech Engines Available on market Each Engines have its own advantages and drawbacks Engines

PC Dictation - Packaged Software

- **Dragon Systems** makes a dictation engine that can be adapted for specific vocabularies.
- **IBM ViaVoice** web site for PC-based dictation application.
- **Lernout & Hauspie** sells a line of PC-based dictation software.
- **FreeSpeech98** is a continuous speech dictation package from Philips Speech Processing for your Windows PC.

PC Recognition Engines

- The **AT&T Advanced Speech Products Group** offers WATSON, with SAPI-compatible speech recognition and speech synthesis, as well as speaker verification technologies.
- **Microsoft Research Speech Technology Group** is developing the Whisper engine.
- **Command Corporation** makes the IN CUBE recognizer.
- SRI Corp's **STAR Lab** has 25 people developing a wide band, continuous speech recognizer called DECIPHER and has been doing some interesting work with a new engine called Corona
- **Apple's Plain Talk** incorporates speech recognition and synthesis into the Mac OS.
- The DDLinux page covers speech engines that have been ported to Linux and even lists a handful of open source engines.
- A variety of PC applications employ dictation engines provided by **IBM** or **Dragon**

Telephony and Call Center Engines

- **Northern Telecom** is developing speech recognition engines.
- **Speech Works** develops speech recognition engine technology for over-the-telephone customer service and transactions.
- **Voice Control Systems** has acquired several other speech recognition companies including Pure Speech. They offers a broad line of telephony-based speech recognition covering digits, cellular voice dialing, auto-attendant, customer service, and transactions, as well as voice verification. They were acquired in May 1999 by Philips Electronics.
- **Nuance** offers a continuous speech recognizer for large vocabulary applications, including stock quotes.
- **Fonix** never came out with its own product but they did acquire the excellent TTS reader AcuVoice and a medical dictation system from Articulate Systems called PowerScribe.
- **Locus Speech Corporation** in Canada has recently announced a speaker-independent, flexible vocabulary telephony speech engine in English and French.
- **Vocalis** focuses on speech for computer telephony in Europe. Started in 1993 by a management buy-out of Logic's Speech and Natural Language Group, they are now a public company based in the UK.

- **Philips Speech Processing** offers leading edge speech recognition, including natural dialogue systems for telephony, consumer products and the IT industry.
- **Lernout & Hauspie** specializes in a broad range of foreign languages and has speech recognition, speech synthesis and speech compression.

2.5 Proposed System

The Drawbacks of existing system are avoided in the proposed system. The proposed system and will be presented in the following form for further analysis.

- Recognizing words
- Text grammar format overview
- Grammar Rules
- Grammar Compiler
- Unicode for Local Language

RECOGNIZING WORDS

Recognizing words is choosing which of all the competitors activated by the bits of acoustic stuff in the utterance accounts best for all those bits. As those bits continue to stream in, listeners must also decide whether they belong to the current word or instead the next word in the utterance; that is, they must also choose among competing divisions of the utterance into words. Various kinds of statistical knowledge (may) play a role in these decisions: acoustical statistics, phoneme

statistics, diphone statistics, syllable statistics, word statistics, "di-word" statistics, etc.

Text grammar format overview

The Extensible Markup Language (XML) format inside a GRAMMAR XML element (block), is an "expert-only-readable" declaration of a grammar that a speech application uses to accomplish the following:

- Improve recognition accuracy by restricting and indicating to an engine what words it should expect.
- Improve translation of recognized speech into application actions. This is made easier by providing "semantic tags," (property name, and value associations) to words/phrases declared inside the grammar.

A GRAMMAR XML element appears in a XML source code file. The XML source is compiled into a binary grammar format and is the format used by SAPI during application runtime.

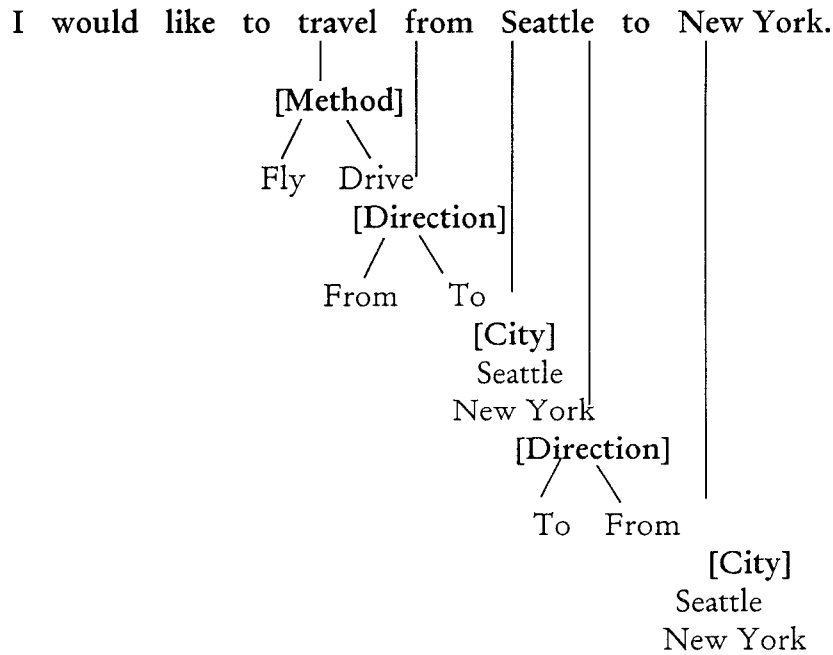
Grammar rules

Grammar rules are elements that SAPI 5.0 compliant recognition engines use to restrict the possible word or sentence choices during the speech recognition process. Recognition engines use grammar rules to control the elements of sentence construction by utilizing the predetermined list of recognized word or phrase

choices. This list of recognized words or phrase choices contained in the grammar rules forms the basis of the recognition engine vocabulary. The phrase or sentence uses each grammar rule element to determine the recognition path.

For example, examine the phrase describing travel plans, **"I would like to travel from Seattle to New York,"** and note that there are elements that determine the resulting information. This is a very simple illustration of what could be a very complex problem. Determining the same travel plans without limiting the method, direction, and travel destination would result in an infinite number of travel options.

The resulting information can be determined by restricting the available choices for a given sentence. Through this method, the resulting information can be composed only from certain available choices, thus eliminating the possibility of an infinite number of travel plan combinations.



`TOPLEVEL`

A grammar tagged as top level can be in an active or inactive state. The rules that import a grammar can override the Activation State of a rule. This conditional state can be configured dynamically at runtime. If an inactive grammar is included in another grammar or grammar rule, ignore the inactive state. When a rule is activated, a speech recognition engine will accept only speech satisfying at least one of the active rules contained in the loaded grammar.

`Non-terminal`

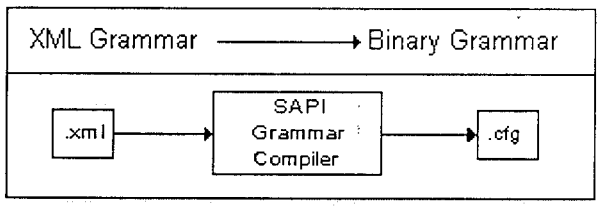
A grammar node is considered to be non-terminal if it is the beginning of a choice selection or a group of choice selections. For example, the grammar node Dog is non-terminal when the subsequent choice selections are types of dogs. This type of grammar is defined as non-terminal because of its choice selections.

`Terminal`

A grammar node is considered to be terminal if it's the only word in the recognized vocabulary, which can be spoken

Grammar Compiler

The SAPI grammar compiler is divided into two parts, the front-end section and the back-end section. The front-end parses the grammars described in XML and optimizes the XML text formatted grammar if requested by the application.



For example, removal of left recursion. The front end then calls the back-end compiler to convert the internal representation into the SAPI binary format.

Unicode for Local Language

The Unicode indic ranges are based on the Indian standard ISCII (Indian Standard Code for Information Interchange, 1988). ISCII is a well thought out standard comprising of all the major Indian scripts. In its Devanagari ranges, it tries to capture all the characters in any Indic Script. For example, while there are no equalent characters in native Devanagari script for letter 'ZHA' which is available in Tamil and Malayalam, (in Unicode, TAMIL LETTER LLLA (U+0BB4) and MALAYALAM LETTER LLLA (U+0D34)), ISCII, and hence Unicode, defines a dummy character for this letter.

Script	Unicode Range	Major Languages
Devanagari	U+0900 to U+097F	Hindi, Marathi, Sanskrit
Bengali	U+0980 to U+09FF	Bengali, Assamese
Oriya	U+0B00 to U+0B7F	Oriya
Tamil	U+0B80 to U+0BFF	Tamil
Telugu	U+0C00 to U+0C7F	Telugu
Kannada	U+0C80 to U+0CFF	Kannada
Malayalam	U+0D00 to U+0D7F	Malayalam

PROGRAMMING

ENVIRONMENT

3. Programming Environment

3.1 System Specification

To develop a Speech Enabling System the following hardware & software were used. Initially users of this system will have to customize the system by pronouncing the language through a micro phone for primary work synchronization.

Hardware Requirements

PROCESSOR	Pentium II
PROCESSOR SPEED	233 MHz
RAM	32 MB
SOUND CARD	Yamaha
INPUTDEVICES	Micro Phones, Speakers

Software Requirements

OPERATING SYSTEM	Windows 98 / win NT		
LANGAUAGE	Visual C++ 6.0,XML		
DEVELOPMENT KIT	Platform Software Development Kit (PSDK)		
	Microsoft Speech SDK 5.0		
	a)	TTS Engine	14.5 Mb
	b)	SR Engine	16 Mb
	c)	Dictation pad (DP)	
INTERFACE	Speech Application Programming Interface (SAPI)		

3.2 Software

Extensible Markup Language

The textual grammar format is an application of the XML. Every XML element consists of a start tag (<SOME_TAG>) and an end tag (</SOME_TAG>) with a case-insensitive tag name and contents between these tags. The start tag and the end tag are the same if the element is empty.

Attributes

Attributes of an XML element appear inside the start tag. Each attribute is in the form of a name followed by an equal sign followed by a string which must be surrounded by either single or double quotation marks. An attribute of a given name may only appear once in a start tag. In summary, the literal string cannot contain either < or ', if the string is surrounded by single quotation marks. It may not contain ", if the string is surrounded by double quotation marks. Furthermore, use all ampersand (&) characters only in an entity reference such as & and >. When a literal string is parsed, the resulting replacement text will resolve all entity references such as > into

its corresponding text, such as. In this specification, only the resulting replacement text needs to be defined for attribute value strings.

Contents

The contents of an element consists of text or sub elements. Formal definitions of valid contents in this specification are provided as regular and "multi-set" expressions. The pseudo-element name "Text" indicates untagged text. With these definitions, the XML specification defines the exact file syntax details.

Comments

The SAPI 5.0 XML parser treats HTML comment tags as unknown XML tag elements. The engine should provide support for comments and other unknown XML elements.

SAPI Utilization

SAPI uses XML content in the following two methods:

- The SAPI context-free grammar compiler, compiles the XML grammar into a binary grammar format. The compiled binary grammar is loaded

into the SAPI runtime environment from a file, memory, or object (.DLL) resource.

- The speech recognition (SR) engine queries the runtime environment for available grammar information.

Definitions

Untagged text declaring a sequence of words that the recognition engine will recognize. Tentatively this text is only the not-necessarily-phonetic representation of words used for reading words whose pronunciation is unknown to the user this form will be called the spelling form.

Non-empty concatenated recognition contents

The contents of a number of XML elements in this specification such as, the P element, contain a sequence of grammar constructs which are concatenated together (one grammar construct after another). These grammar elements must be recognized in order for the contents defined to be recognized.

Microsoft
Speech SDK

Microsoft Speech SDK

The Microsoft Speech API (SAPI) is a software layer allowing speech-enabled applications to communicate with speech engines (Speech Recognition (SR) engines and Text to Speech (TTS) engines). SAPI includes an Application Programming Interface (API) and a Device Driver Interface (DDI). Applications communicate with SAPI via the API layer and speech engines communicate with SAPI via the DDI layer.

A speech-enabled application and an SR engine do not directly communicate with each other – all communication is done via SAPI. SAPI takes responsibility for a number of aspects of a speech system, such as:

- Controlling audio input, whether from a microphone, files, or a custom audio source; and converting audio data to a valid engine format.
- Loading grammar files, whether dynamically created or created from memory, URL or file; and resolving grammar imports and grammar editing.
- Compiling standard SAPI XML grammar format, and conversion of custom grammar formats, and parsing semantic tags in results.
- Sharing of recognition across multiple applications using the shared engine. All marshalling between engine and applications is done by SAPI.

- Returning results and other information back to the application and interacting with its message loop or other notification method. This allows an engine to have a much simpler threading model than in SAPI 4, as SAPI 5 does much of the thread handling.
- Storing audio and serializing results for later analysis.
- Ensuring that applications do not cause errors – preventing applications from calling the engine with invalid parameters, and dealing with applications hanging or crashing.

SAPI Architecture

SAPI model is divided into two distinct levels:

High-level SAPI

This level provides basic speech services in the form of command-and-control speech recognition and simple text-to-speech output.

This level interface has two top-level objects-

1.Voice Command Object

- **Voice Command interface** is used to enumerate, create, and delete voice menu objects. This interface is also used to register an application to use the SR engine.

- **Attributes interface** is used to set and retrieve a number of basic parameters that control the behavior of the voice command system. You can enable or disable voice commands, adjust input gain, establish the SR mode, and control the input device
- **Dialogs interface** gives you access to a series of dialog boxes that can be used as a standard set of input screens for setting and displaying SR engine information

2 Voice Text Object

- **Voice Text interface** provides a set method to start, pause, resume, fast forward, rewind, and stop the TTS engine while it is speaking text.
- **Attribute interface** provides playback speed (in words per minute), speaking mode (male, female, child, adult, and so on).
- **Dialogs interface** can be used to allow users the ability to set and retrieve information regarding the TTS engine

Low-level SAPI

This level provides detailed access to all speech services, including direct interfaces to control dialogs and manipulation of both speech recognition (SR) and text-to-speech (TTS) behavior attributes.

3.3 Description of Software Tools used

MS SPEECH SDK	*Microsoft Speech Software Development Kit*
PSDK	Platform Software Development Kit
TTS	Text to Speech Engine
SRE	Speech Recognition Engine
DSR	Direct Speech Recognition
SAPI	Speech Application Programming Interface
DP	Dictation pad

SYSTEM DESIGN

&

DEVELOPMENT

4. System Design and Development

4.1 Input Design

The application should provide the user given input through our voice. The style of input is established during software requirements analysis and Design input styles will vary the degree of human interaction. The inputs to the system are fully voice. Sequentially, we have to give the input. The first thing the application has to do is to choose the Input Media where the SR will occur. It does so by selecting a Speaker as an Input media. The application then selects a voice for Speech Recognition Engine that suits its needs, with the language corresponding to Speech to text The SR engine is then connected to the desired Input Media. The application can begin to convert speak to text by sending voice to the SR engine. As you can see all this seems to be very simple but one of the main point is to connect the engine to audio hardware. At this time Microsoft provides audio destination objects for wave file input and for multimedia devices. speech recognition system uses four key operations to listen to and understand human speech. They are:

- **Word separation**-This is the process of creating discreet portions of human speech. Each portion can be as large as a phrase or as small as a single syllable or word part.
- **Vocabulary**-This is the list of speech items that the speech engine can identify.
- **Word matching**-This is the method that the speech system uses to look up a speech part in the system's vocabulary-the *search engine* portion of the system.
- **Speaker dependence**-This is the degree to which the speech engine is dependent on the vocal tones and speaking patterns of individuals.

Finally, the speech engine must balance all of these factors against the aspect of speaker dependence. As the speech system learns smaller and smaller differences between words, the system becomes more and more dependent on the speaking habits of a single user. Individual accents and speech patterns can confuse speech engines. In other words, as the system becomes more responsive to a single user, that same system becomes less able to translate the speech of other users.

4.2 Output Design

The output is the essential element of any system. As the output can be presented in many different ways, the appropriate method should be used while presenting the user with the output that is convenient for them. So the output is the prime element to be designed in the system. As the application itself is a report writing utility, reports have to be designed. Also dialogue which is the main form of interaction between the system and the user, should be friendly and must provide the after information if any in a convenient way.

4.3 Process Design

Recognized raw audio involves the following Process

1. **Frequency Analysis**

   Raw audio Involved in to Various Digital Signal process In the Digital Signal process in this process we can use Various Mathematical Algorithm like Pretty Hairy mathematical Algorithms, Fourier Transforms

2. **Recognize Phonemes**

   Recognize the Pronunciation and broken up in to Phonemes(Pronunciation Dictionary)

   Example: - Original - The man Walked

   Phonemes - Th-uh M-a-Nw-au-l-k-tD

3. **Sound to phoneme Database**

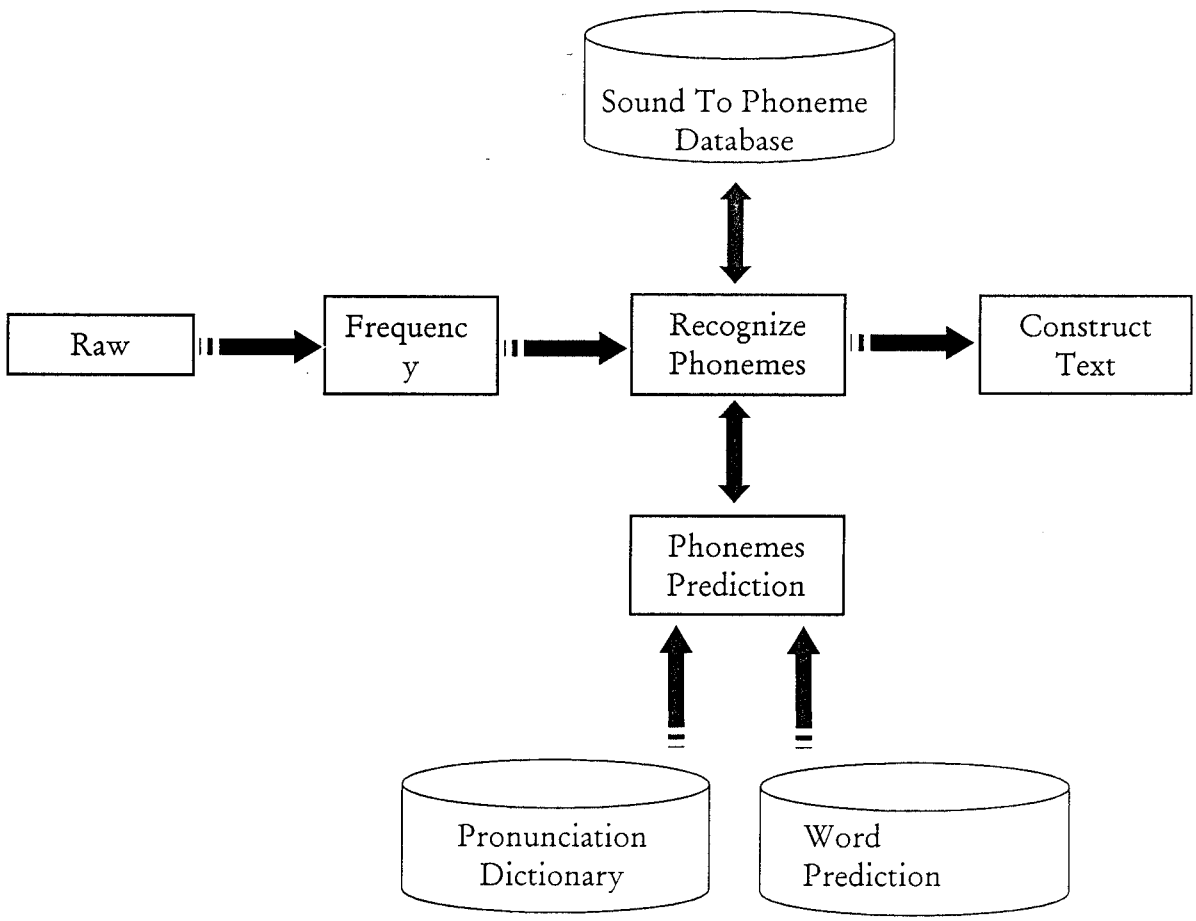
   First broken Phonemes matched in to particular sound Database and converted to Text. But this process takes long time

4. **Phoneme Prediction**

   Predict which phonemes are likely to occur the particular context

   For example ' f ' sound never comes before ' s '

Pictorial view



5. Word Prediction

Further reduce the phoneme candidate list

Example :-

Recognizer will listen ' s ' and ' h ' but the word

Prediction database says " yes " or " yeh "both are same

SYSTEM TESTING

&

IMPLEMENTATION

5. System Testing and Implementation

5.2 System Testing

Testing is an activity that ensures that a correct system is built. It is restricted to being performed after the development phase is completed. But it is to be carried out in parallel with all the stages of system development. Software testing has been carried out by using the existing data and found to be matching well.

Testing Objectives

There are several rules that can serve as testing objectives. They are,

- Testing is a process of executing a program with the intention of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objection stated above, it could uncover errors in the software.

Testing has become an integral part of any system or project. When the software is developed, before it is given to the user it must be tested whether it is solving the purpose for which it is developed.

Testing is the process where the test data is prepared and is used for testing the modules individually and later validated.

The following is the description of testing strategies, which are carried out during the testing period. Code testing examines the logic of the program. The module testing, to locate error in each module is tested individually. This enables us to detect errors and correct it without affecting other modules. When the user finds no major problems with its accuracy the system passes to a final acceptance test. This test confirms that the system meets the original objectives, goals and requirements.

5.2 System Implementation

Implementation of Local Language Engine

SAPI provides a means for applications to set certain settings and configurations on the SR engine. You want to implement local languages Engine means go threw the following steps

1.Set the Sample Engine to the Default Engine

- In Control Panel, double-click the Speech icon.
- On the Speech Recognition tab, select the SAPI 5.0 Sample Engine from the list of available engines.
- Sample Engine Basic Behavior

  The sample engine randomly generates context free grammar (CFG) recognition results based on the CFG grammar you select. The sample engine will also generate other events, such as interference and requestui etc.

The engine does not perform the recognition based on an acoustic or language model. Instead, it retrieves the CFG grammar information from SAPI and constructs random results.

2. Create the .wav files

According to the new string table and place this under the specified directory '..\resources',

3. Grammar Creation

Each CFG grammar contains one or more *rules*. Rules can be *top-level*, indicating that they can be *activated* for recognition. Each rule has an *initial state* and additional *states*, connected by *transitions*. Each transition can be one of several types:

- A *word transition* indicating a word to be recognized
- A *rule transition* indicating a reference to a *sub-rule*
- An *epsilon* (null) transition
- Some *special transitions* for such features as embedding dictation within a CFG.

Sample XML Grammar

```
<GRAMMAR>
  <RULE NAME="TOPLEVEL">
    <P> வணக்கம் </P>
  </RULE>
</GRAMMAR>
```

Create and compile the appropriate XML files using a grammar editor and compiler.

4. **Include the CFG binaries**

Include the .dll by importing the CFG file names into srcomp.rc

5. **Recompile the Source**

Using VC++ to recompile project

The implementation of the system in the organization comes as the last Step of the system development. This phase of system consist of

- Testing the document programs with the sample input.
- Correction of any errors identified.
- Making necessary changes to the system with the actual data.
- Performing a parallel run of the system and find out any errors
- of calculation present and correct to them.
- Training the user personnel.

As a part of the training program have made a seminar about the system To the user group on the systems working. The users were found to be Satisfied with the system and now trying to study the new system.

CONCLUSION

6. Conclusion

The combination of speech recognition and English Query represents a powerful way for a user to access information in a SQL Server database very quickly. For users who work in an environment where speed and ease of access are critical, it holds enormous promise for future applications. As hardware continues to become more powerful and cheaper, speech recognition should continue to become more accurate and useful to increasingly wider

Soon, speech technology will allow people to access Web content from a cellular telephone and connect to their corporate networks securely over smart wireless devices. Speech-enabled devices, networks, and Web sites will provide a new and lucrative market for software and hardware developers, and they will help wireless operators generate greater service revenues.

FUTURE SCOPE

7. Future Enhancements

There are enormous practical utilities and advantages with voice user interface between user and machine. These systems with speech recognition could be used to automate the industries, there by reducing the man power to minimum. A speech is the most natural and spontaneous way of communication with humans, it would be extremely advantages to have speech communication with the machine. The data could be entered in to the computer through the speech communication between man and computer. Such communications increase the convenience of information transfer from man to machine. Isolated word recognizer could be used at 20 to 100 word per minute.(compared to 15 words per minute for a slow typist and about 45 words for a fast one),While continuos speech recognizers could operate at 150 to 200 words per minute handicapped person could operate various devices by spoken command, making him less dependent on others. Speech recognition systems could be used for security purpose and it is also used for language translators.

Microsoft SAPI 5.0 Enhancements

Full COM support (such as UNICODE and self registering DLLs);

- New DDI model for handling engine processing
- Application support for shared speech recognizer

System-wide management of objects including SR engines, TTS voices, audio devices, and lexicons

New voice object for handling audio formats and barge-in; voice handles both synthetic and recorded audio through same interface

Improved free-threading model for system fault tolerance

Handles all application interaction, parameter validation, and bad-application cleanup

Standard mechanism for invoking and managing user interfaces, such as user enrollment and microphone setup

Applications can retain audio in any ACM format regardless of original format

Use of alternates for corrections

CFG namespace support for XML-based grammar format & load from Internet

CFG semantic tags allow nested properties, eliminating reparsing of recognized text

SAPI 3.0, 4.0a & 5.0 can coexist on the same machine

Consumer Speech Portal: A woman driving to the airport uses her mobile phone to call a speech-controlled Web portal. By speaking into the phone, she can check her flight status, listen to the latest business news, sell stock, and hear the weather forecast for her destination city.

Enterprise Speech Portal: A corporate sales representative on his way to meet a client uses his cell phone to call into his company's network and check his e-mail, voice mail, and calendar, retrieve the client's address, and check the Customer Resource Management system to find background information on the client.

Speech Customer Service: A system administrator at an off-site training seminar needs to check the status of computers she ordered for her company. She uses her phone to call the PC-manufacturer's speech-enabled Web site and ask for the information she needs about the order.

Speech Commerce: A consumer uses a smart phone to access an online bookstore. Rather than pushing tiny buttons he simply says, "Show me the latest bestsellers." When the books appear, he can buy one with a natural-language speech command, such as "I want to buy this book." The Web site confirms his request by speaking it back to him.

Speech-enabled applications. Microsoft is developing solutions that will give end users access to corporate and Web-based information from any mobile phone simply by speaking. Speech-enabled applications include e-mail and personal assistant, as well as mobile portal content and customer services. An overall speech shell will give callers a single point of access and consistent interface to all speech-enabled applications Mobile carriers will enable companies to offer employees these services securely over the Internet.

Speech page development. Microsoft is actively working with standards bodies to develop XML techniques for defining speech-enabled dialog interactions. When coupled with Microsoft's powerful graphical development tools, this will open the speech-enabled application industry to wireless developers, independent software vendors, and corporate developers. Speech Gateways will connect to the voice network and "browse" XML-based applications, or "speech pages," allowing end users to access a broad range of speech-enabled services through a single point of access.

Speech Gateway. Microsoft is developing a Speech Gateway for service providers. It will offer a single speech interface to all applications and services. A speech shell application will authenticate callers and offer them a customized range of services. The speech interfaces to these services are defined by speech pages. The Speech Gateway works together with a carrier wireless access server that can render the same core services to a range of Web-enabled devices with screen and keypad interfaces.

Corporate technologies. Microsoft is working to provide companies with a corporate wireless access server that makes data on the corporate network available securely in real-time to mobile users. The server would act as the secure gateway between the corporate network and the wireless public network. Companies will also be able to use a speech access client to offer their employees access to corporate Microsoft Exchange-based and other data from any telephone, via the wireless service provider's speech gateway.

Speech Web Benefits adding speech-enabled access to networked information offers a number of advantages for both mobile operators and corporations.

Mobile operator benefits Speech portal services deliver a clear business benefit for mobile operators. Subscribers will pay for more voice minutes because they'll use their wireless service to accomplish many tasks in addition to making standard telephone calls. Speech technology can bring the Web to virtually all mobile phone users immediately. End users don't need to buy or use any new hardware or software.

Adding a speech interface to personalized value-added services will increase customer loyalty and reduce customer churn.

Carriers can define and offer a wide range of speech-enabled services using the same speech gateway platform and a single speech-shell user interface, accessible from a single number.

By using an industry-standard XML markup language as the basis for speech-enabled Web pages, service providers can easily add speech access to their core Web services. The operator's speech gateway can link to speech pages offered by third-party content partners or corporate customers using the same XML standards.

Powerful tools for developing speech pages put the power to define and differentiate speech services in the operator's hands.

Corporate Benefits Knowledge workers can access corporate messages, appointment data, contacts, and data with no more equipment than a cellular phone.

By working with a mobile operator with a wireless access platform, corporations can provide secure wireless access without investing in additional server or telephony hardware or phone lines.

Corporations can add speech interfaces to existing Internet or intranet applications by developing customized, XML-based speech pages.

BIBILOGRAPHY

Bibliographic References

Research Articles

Investemnt on Infrastructure defined in appendix

Automatic Detection of Poor Speech Recognition at the Dialogue Level
Diane J. Litman, Marilyn A. Walker and Michael S. Kearns
AT&T Labs Research
180 Park Ave, Bldg 103
Florham Park, N.J. 07932

mkearns@research.att.com
diane@research.att.com
walker@research.att.com

Characterizing and Recognizing Spoken Corrections in Human- Computer Dialogue by Gina-Anne Levow
MIT AI Laboratory
Room 769, 545 Technology Sq
Cambridge, MA 02139

gina@ai.mit.edu

Reference Books

- Aderman, D. & Smith, E.E. (1971) Expectancy as a determinant of functional units in perceptual recognition. *Cognitive Psychology*, 2:117-129

- Web Smith, B. (1988), "Materials Towards a History of Speech Act Theory", in Eschbach, A. (ed.), *Karl Bühler's Theory of Language*. Amsterdam, 125-52. References

- Becker, C.A. (1980) Semantic context effects in visual word recognition: An analysis of semantic strategies. *Memory & Cognition* , 8:493-512.

- Software engineering by Shooman.

- Visual C++ Programming by YashavantKanetkar

- Software engineering by Pressman.

- Software Project management by Richard Flair.

Web References

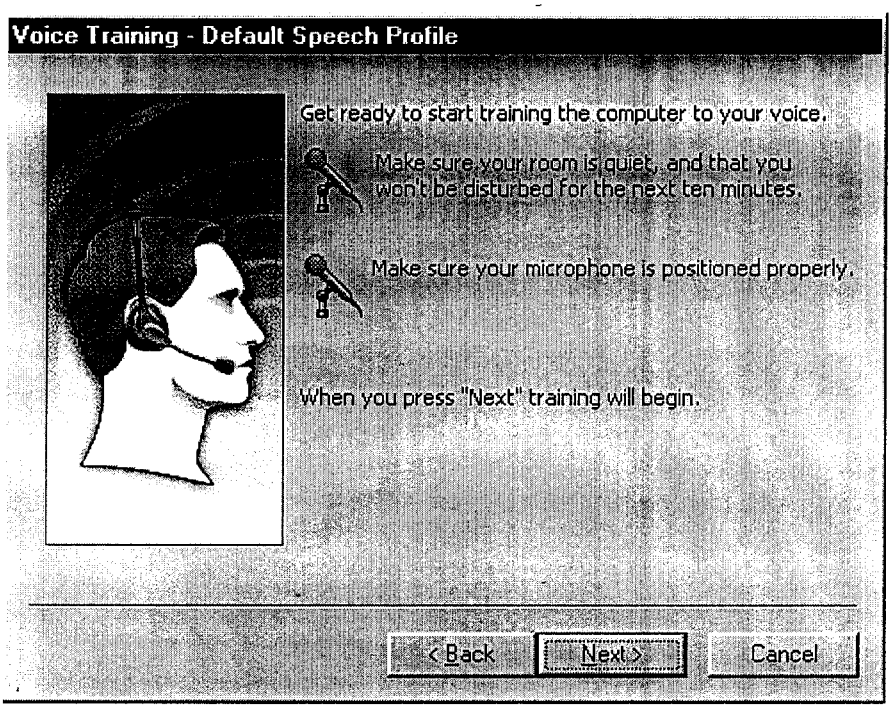
http://www.mailgate.org/comp/comp.speech.research/
http://citeseer.nj.nec.com/Applications/SpeechRecognition/date.html
http://www.cs.cmu.edu/afs/cs/user/robust/www/papers.html
http://citeseer.nj.nec.com/433532.html
http://developer.apple.com/samplecode/Sample_Code/Sound/Speech_Recognition_Sample.htm
http://www.edc.org/spk2wrt/hypermail/
http://www.dfki.de/etai/SpecialIssues/Dia99/koelzer/html/
http://www.tiac.net/users/rwilcox/speech.html#RECO
http://www.microsoft.com/business/mobility/speech.asp
http://www.zdnet.co.uk/pcmag/supp/1998/speech/3.html
http://www.generation5.org/aisolutions/sr00.shtml
http://www.escription.com/editscript_for_editing_medical_transcriptions.htm
http://www.ee.ualberta.ca/~elliott/ee552/projects/1999f/VRremote/vrfinal.htm
http://www.nortelnetworks.com/solutions/eba/fs/speech.html
http://www.manufacturing.net/magazine/dn/archives/2000/dn0117.00/02f1985.htm
http://www.ti.com/sc/docs/psheets/abstract/apps/spra317.htm
http://ii2.ai.iit.nrc.ca/VoiceCode/whitepaper.html
http://sourceforge.net/projects/voicecode/

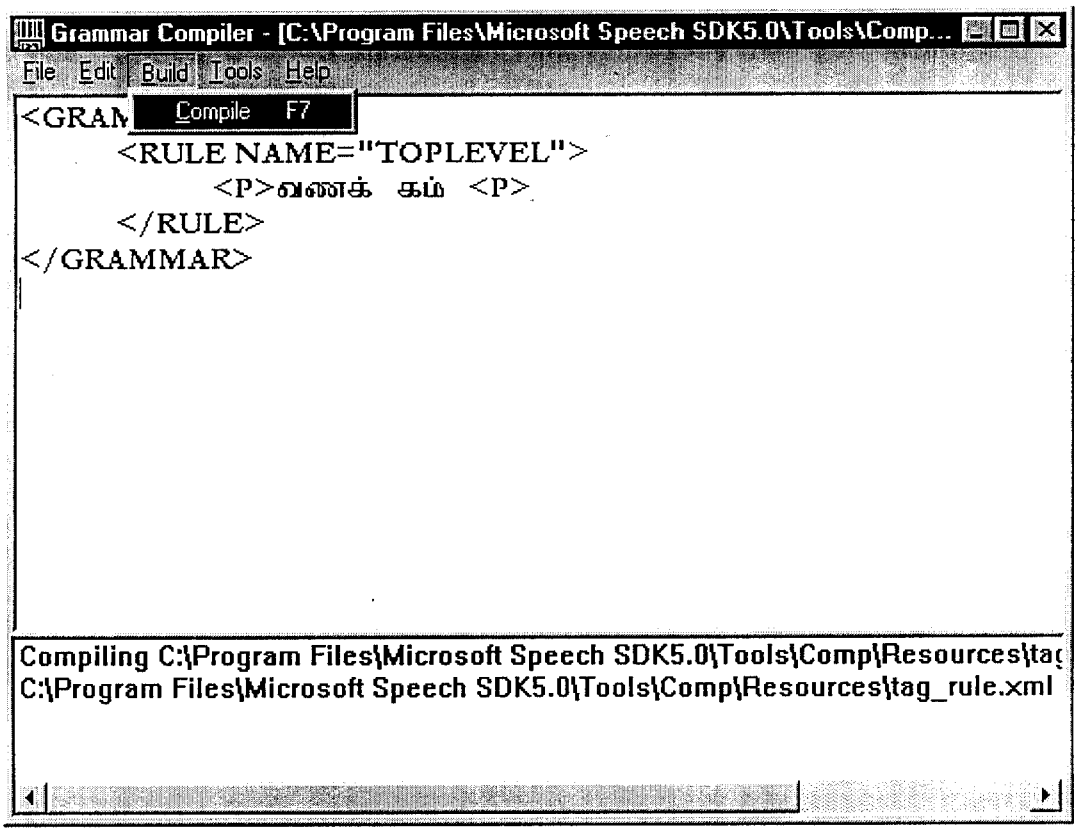
http://www.sunny-beach.net/vbvoicetelephony.htm
http://www.speech.cs.cmu.edu/comp.speech/
http://www.speech.cs.cmu.edu/comp.speech/FAQ.Contents.html
http://www.voicexml.org/
http://www.synapseadaptive.com/
http://www.isip.msstate.edu/projects/speech/software/index.html
http://msdn.microsoft.com/library/periodic/period99/ppt2000.htm
http://www.research.microsoft.com/srg/docs/CVoiceText.html

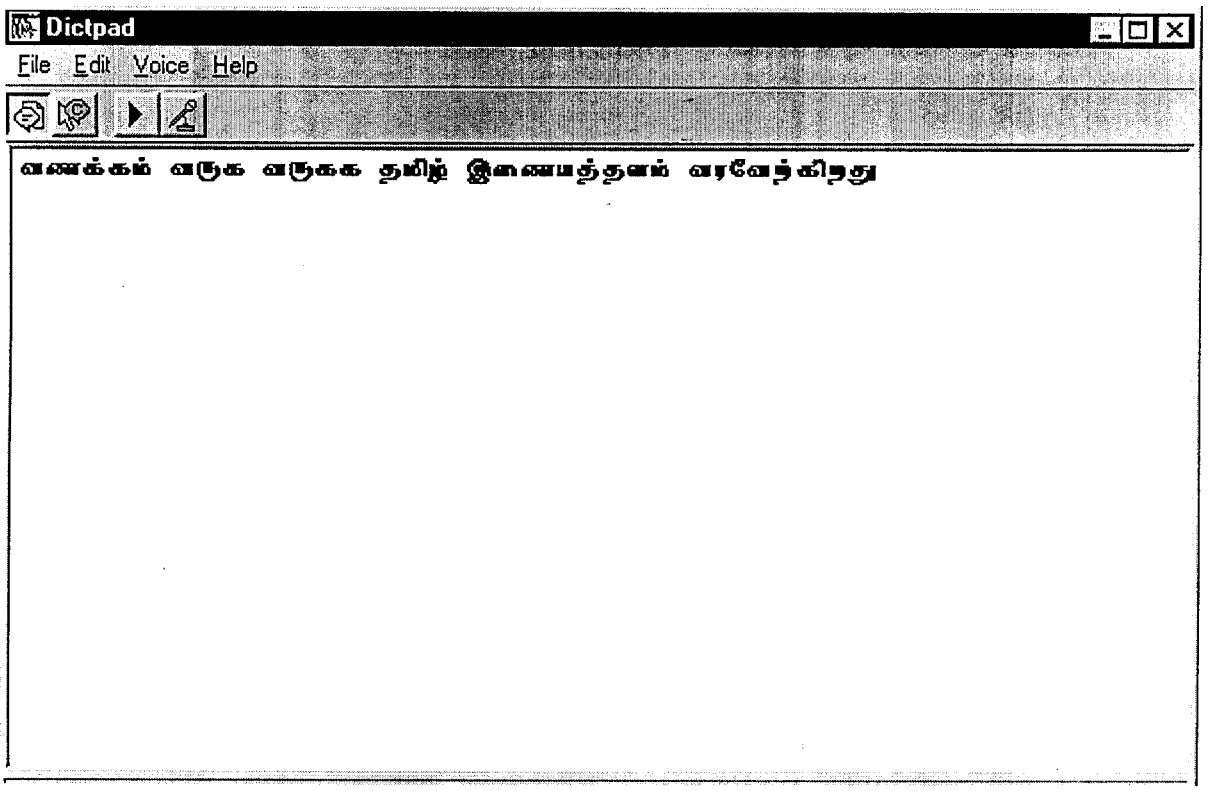
APPENDIX – A

SCREENS

Screens


Voice Training - Default Speech Profile
Get ready to start training the computer to your voice.
Make sure your room is quiet, and that you won't be disturbed for the next ten minutes.
Make sure your microphone is positioned properly.
When you press "Next" training will begin.
< Back
Next >
Cancel


Grammar Compiler - [C:\Program Files\Microsoft Speech SDK5.0\Tools\Comp...
File Edit Build Tools Help
Compile F7
<GRAM
<RULE NAME="TOPLEVEL">
<P>வணக் கம் <P>
</RULE>
</GRAMMAR>
Compiling C:\Program Files\Microsoft Speech SDK5.0\Tools\Comp\Resources\tag
C:\Program Files\Microsoft Speech SDK5.0\Tools\Comp\Resources\tag_rule.xml


Dictpad
File Edit Voice Help
வணக்கம் வருக வருக தமிழ் இணையத்தளம் வரவேற்கிறது

APPENDIX – B

RESEARCH ARTICLES

Appendix- B
Research
Articles

Automatic Detection of Poor Speech Recognition at the Dialogue Level

Diane J. Litman, Marilyn A. Walker and Michael S. Kearns
AT&T Labs Research
180 Park Ave, Bldg 103
Florham Park, N.J. 07932
diane,walker,mkearns
@research.att.com

Abstract

The dialogue strategies used by a spoken dialogue system strongly influence performance and user satisfaction An ideal system would not use a single fixed strategy, but would *adapt* to the circumstances at hand. To do so, a system must be able to identify dialogue properties that suggest adaptation. This paper focuses on identifying situations where the speech recognizer is performing poorly. We adopt a machine learning approach to learn rules from a dialogue corpus for identifying these situations. Our results show a significant improvement over the baseline and illustrate that both lower-level acoustic features and higher-level dialogue features can affect the performance of the learning algorithm.

1 Introduction

Builders of spoken dialogue systems face a number of fundamental design choices that strongly influence both performance and user satisfaction. Examples include choices between user, system, or mixed initiative, and between explicit and implicit confirmation of user commands. An ideal system wouldn't make such choices *a priori*, but rather would *adapt* to the circumstances at hand. For in-stance, a system detecting that a user is repeatedly uncertain about what to say might move from user to system initiative, and a system detecting that speech recognition performance is poor might switch to a dialogue strategy with more explicit prompting, an explicit confirmation mode, or keyboard input mode. Any of these adaptations might have been appropriate in dialogue D1 from the Annie sys-tem (Kamm et al., 1998), shown in Figure 1.In order to improve performance through such adaptation, a system must first be able to identify, in real time, salient properties of an ongoing dialogue that call for some useful change in system strategy. In other words, adaptive systems should try to *automatically* identify *actionable* properties of ongoing dialogues. Previous work has shown that speech recognition performance is an important predictor of user satisfaction, and that changes in dialogue behavior impact speech recognition performance (Walker et al., 1998b; Litman et al., 1998; Kamm et al., 1998). Therefore, in this work, we focus on the task of automatically detecting poor speech recognition performance in several spoken dialogue systems developed at AT&T Labs. Rather than hand-crafting rules that classify speech recognition performance in an ongoing dialogue, we take a machine learning approach. We begin with a collection of system logs from actual dialogues that were labeled by humans as having had "good" or

"bad" speech recognition (the training set). We then apply standard machine learning algorithms to this training set in the hope of discovering, in a principled manner, classifiers that can automatically detect poor speech recognition during novel dialogues. In order to train such classifiers, we must provide them with a number of "features" of dialogues de-rived from the system logs that might allow the sys-tem to automatically identify poor recognition performance. In addition to identifying features that provide the best quantitative solutions, we are also interested in comparing the performance of classifiers derived solely from acoustic features or from "high-level" dialogue features, and from combinations of these and other feature types. Note that we are free to invent as many features as we like, as long as they can be computed in real time from the raw system logs. Since the dialogue systems we examine use automatic speech recognition (ASR), one obvious feature available in the system log is a per-utterance score from the speech recognizer representing its "confidence" in its interpretation of the user's utterance (Zeljkovic, 1996). For dialogue D1, the recognizer's output and the associated confidence scores

Figure 1: Example dialogue excerpt D1 with Annie.

Actual Dialogue		*ASR Interpretation*	*ASR Confidence*
A1	Annie here.		
U1	Hello Annie, could you give me the information on a meeting I have tomorrow?	REJECT	-4.84
A2	What was that?		
U2	I need to find out uh, a meeting time and place. Kim sent me a message on it	REJECT	-4.20
A3	For all options say help me out		
U3	Help me out	REJECT	-6.35
A4	Annie here		
U4	Help.	Help.	-1.74
A5	Help. Say call or say send a message, followed by a label or a number Or say get my messages	help	
U5	Get messages*(user interrupts A5*	get me sips	-3.33
A6	Speech and Image Processing Lab.		
U6	Help	help	-1.78
A7	Help. Say the first and last name ...		
U7	Kim.	Annie	-3.81

are in the last two columns of Figure 1. These confidence measures are based on the recognizer's language and acoustic models. The confidence scores are typically used by the system to decide whether it believes it has correctly understood the user's utterance. When the confidence score falls below a threshold defined for each system, the utterance is considered a rejection (e.g., utterances U1, U2, and U3 in D1). Note that since our classification problem is defined by speech recognition performance, it might be argued that this confidence feature (or features derived from it) suffices for accurate classification. However, an examination of the transcript in D1 suggests that other useful features might be derived from global or high-level properties of the dialogue history, such as features representing the system's repeated use of diagnostic error messages (utterances A2 and A3), or the user's repeated requests for help (utterances U4 and U6). Although the work presented here focuses exclusively on the problem of automatically *detecting* poor speech recognition, a solution to this problem clearly suggests system *reaction*, such as the strategy changes mentioned above. In this paper, we re-port on our initial experiments, with particular attention paid to the problem definition and methodology, the best performance we obtain via a machine learning approach, and the performance differences between classifiers based on acoustic and higher-level dialogue features.

2 Systems, Data, Methods

The learning experiments that we describe here use the machine learning program RIPPER (Co-hen,1996) to automatically induce a "poor speech recognition performance" classification model from a corpus of spoken dialogues.1 RIPPER (like other learning programs, such as C5.0 and CART) takes as input the names of a set of *classes* to be learned, the names and possible values of a fixed set of *features*, *training data* specifying the class and feature values for each example in a training set, and out-puts a *classification model* for predicting the class of future examples from their feature representation. In RIPPER, the classification model is learned using greedy search guided by an information gain metric, and is expressed as an ordered set of if-then rules. We use RIPPER for our experiments because it sup-ports the use of "set-valued" features for representing text, and because if-then rules are often easier for people to understand than decision trees (Quinlan,1993). Below we describe our corpus of dialogues, the assignment of classes to each dialogue, the extraction of features from each dialogue, and our learning experiments.

Corpus: Our corpus consists of a set of 544 dialogues (over 40 hours of speech) between humans and one of three dialogue systems: ANNIE (Kammet al., 1998), an agent for voice dialing and messaging; ELVIS (Walker et al., 1998b), an agent for accessing email; and TOOT (Litman and Pan,1999), an agent for accessing online train schedules. Each agent was implemented using a general-purpose platform for phone-based spoken dialogue systems (Kamm et al.,1997). The dialogues were obtained in controlled experiments designed to evaluate dialogue strategies for each agent. The exper-1 We also

ran experiments using the machine learning pro-gram BOOSTEXTER (Schapire and Singer, To appear), with results similar to those presented below. Elements required users to complete a set of application tasks in conversations with a particular version of the agent. The experiments resulted in both a digitized recording and an automatically produced sys-tem log for each dialogue.

Class Assignment: Our corpus is used to construct the machine learning classes as follows. First, each utterance that was not rejected by automatic speech recognition (ASR) was manually labeled as to whether it had been semantically misrecognize or not.2 This was done by listening to the recordings while examining the corresponding system log. If the recognizer's output did not correctly capture the task-related information in the utterance, it was labeled as a misrecognition. For example, in Figure 1 U4 and U6 would be labeled as correct recognition's,

while U5 and U7 would be labeled as misrecognitions.

Note that our labeling is semantically based; if U5 had been recognized as "play messages"(which invokes the same application command as "get messages"), then U5 would have beenlabeled as a correct recognition. Although this labeling needs to be done manually, the labeling is based on objective criteria. Next, each dialogue was assigned a class of either *good* or *bad*, by thresholding on the percentage of user utterances that were labeled as ASR semantic misrecognitions. We use a threshold of 11% to balance the classes in our corpus, yielding 283 good and 261 bad dialogues.3 Our classes thus reflect *relative* goodness with respect to a corpus. Dialogue D1 in Figure 1 would be classified as "bad", be-cause U5 and U7 (29% of the user utterances) are misrecognized.

Feature Extraction: Our corpus is used to construct the machine learning features as follows. Each dialogue is represented in terms of the 23 primitive features in Figure 2. In RIPPER, feature values are continuous (numeric), set-valued, or symbolic. Feature values were automatically computed from system logs, based on five types of knowledge sources: acoustic, dialogue efficiency, dialogue quality, experimental parameters, and lexical. Previous work correlating misrecognition rate with acoustic information, as well as our own 2 These utterance labeling were produced during a previous set of experiments investigating the performance evaluation of spoken dialogue systems (Walker et al., 1997; Walker et al., 1998a; Walker et al., 1998b; Kamm et al., 1998; Litman et al.,1998; Litman and Pan, 1999). This threshold is consistent with a threshold inferred from human judgements (Litman, 1998).

Figure 2: Features for spoken dialogues.

Acoustic Features

mean confidence, pmisrecs%1, pmisrecs%2, pmis-recs%3, pmisrecs%4

Dialogue Efficiency Features

elapsed time, system turns, user turns

Dialogue Quality Features

rejections, timeouts, helps, cancels, bargeins (raw) rejection%, timeout%, help%, cancel%, bargein% (nor-malized)

Experimental Parameters Features

system, user, task, condition

Lexical Features

ASR text

The acoustic, dialogue efficiency, and dialogue quality features are all numeric-valued. The acoustic features are computed from each utterance's confidence (log-likelihood) scores (Zeljkovic,1996). *Mean confidence* represents the average log-likelihood score for utterances not rejected during ASR. The four *pmisrecs%* (predicted percentage of misrecognitions) features represent different (coarse) approximations to the *distribution* of log-likelihood scores in the dialogue. Each *pmis-recs%* feature uses a fixed threshold value to predict whether a non-rejected utterance is actually a misrecognition, then computes the percentage of user utterances in the dialogue that correspond to these *predicted* misrecognitions. (Recall that our dialogue

classifications were determined by thresholding on the percentage of *actual* misrecognitions.) For in-stance, *pmisrecs%1* predicts that if a non-rejected utterance has a confidence score below then it is a misrecognition. Thus in Figure 1, utterances U5 and U7 would be predicted as misrecognitions using this threshold. The four thresholds used for the four *pmisrecs%* features are -1,-3,-4,-5, and were chosen by hand from the entire data set to be informative.

The dialogue efficiency features measure how quickly the dialogue is concluded, and include *elapsed time* (the dialogue length in seconds), and *system turns* and *user turns* (the number of turns for each dialogue participant).

Figure 3: Feature representation of dialogue D1.

Mean confidence	pmisrecs %1	pmisrecs %2	pmisrecs %3	pmisrecs %4	Elapsed time	system turns	user turns
-2.7	29	29	0	0	300	7	7

The dialogue quality features attempt to capture aspects of the naturalness of the dialogue. *Rejections* represents the number of times that the sys-tem plays special rejection prompts, e.g., utterances A2 and A3 in dialogue D1. This occurs whenever the ASR confidence score falls below a threshold associated with the ASR grammar for each system state (where the threshold was chosen by the system designer). The *rejections* feature differs from the *pmisrecs%* features in several ways. First, the *pmis-recs%* thresholds are used to determine misrecognitions rather than rejections. Second, the *pmisrecs%* thresholds are fixed across all dialogues and are not dependent on system state. Third, a system rejection event directly influences the dialogue via the rejection prompt, while the *pmisrecs%* thresholds have no corresponding behavior. *Timeouts* represents the number of times that the system plays special timeout prompts because the user hasn't responded within a pre-specified time frame. *Helps* represents the number of times that the system responds to a user request with a (context-sensitive) help message. *Cancels* represents the number of user's requests to undo the system's previous action. *Bargeins* represents the number of user attempts to interrupt the system while it is speaking.4 In

addition to raw counts, each feature is represented in normalized form by expressing the feature as a percentage. For example, *rejection%* represents the number of rejected user utterances divided by the total number of user utterances. In order to test the effect of having the maximum amount of possibly relevant information available, we also included a set of features describing the experimental parameters for each dialogue (even though we don't expect rules incorporating such features to generalize). These features capture the conditions under which each dialogue was col-4 Since the system automatically detects when a bargain occurs, this feature could have been automatically logged. However, because our system did not log bargains, we had to hand-label them. The experimental parameters features each have a different set of user-defined symbolic values. For example, the value of the feature *system* is either "annie", "elvis", or "toot", and gives RIPPER the option of producing rules that are system-dependent. The lexical feature *ASR text* is set-valued, and represents the transcript of the user's utterances as output by the ASR component.

Learning Experiments: The final input for learning is training data, i.e., a representation of a set of dialogues in terms of feature and class values. In order to induce classification rules from a variety of feature representations our training data is represented differently in different experiments. Our learning experiments can be roughly categorized as follows. First, examples are represented using all of the features in Figure 2 (to evaluate the optimal level of performance). Figure 3 shows how Dialogue D1 from Figure 1 is represented using all 23 features. Next, examples are represented using only the features in a single knowledge source (to comparatively evaluate the utility of each

knowledge source for classification), as well as using features from two or more knowledge sources (to gain insight into the interactions between knowledge sources). Finally, examples are represented using feature sets corresponding to hypotheses in the literature (to empirically test theoretically motivated proposals). The output of each machine learning experiment is a classification model learned from the training data. To evaluate these results, the error rates of the learned classification models are estimated using the re sampling method of *cross-validation* (Weiss

and Kulikowski, 1991). In 25-fold cross-validation, the total set of examples is randomly divided into 25 disjoint test sets, and 25 runs of the learning pro-gram are performed. Thus, each run uses the examples not in the test set for training and the remaining examples for testing. An estimated error rate is obtained by averaging the error rate on the testing

3 Results

Summarizes our most interesting experimental results. For each feature set, we report accuracy rates and standard errors resulting from cross validation.5 It is clear that performance depends on the features that the classifier has available. The BASELINE accuracy rate results from simply choosing the majority class, which in this case means predicting that the dialogue is always "good". This leads to a 52% BASELINE accuracy. The *REJECTION%* accuracy rates arise from a classifier that has access to the percentage of dialogue utterances in which the system played a rejection message to the user. Previous research suggests that this acoustic feature predicts misrecognitions because users modify their pronunciation in response to system rejection messages in such a way as to lead to further misunderstandings However, despite our expectations,

The *REJECTION%* accuracy rate is not better than the BASELINE at our desired level of statistical significance. Using the EFFICIENCY features does improve the performance of the classifier significantly above the BASELINE (61%). These features, however, tend to reflect the particular experimental tasks that the users were doing. The EXP-PARAMS (experimental parameters) features are even more specific to this dialogue corpus than the efficiency features: these features consist of the name of the system, the experimen-5 Accuracy rates are statistically significantly different when the accuracies plus or minus twice the standard error do not overlap Since with the exception of the

experimental condition these features are specific to this corpus, we wouldn't expect them to generalize.

Figure 4: EXP-PARAMS rules.

if (condition = mixed) then *bad*

if (system = toot) then *bad*

if (condition = novices without tutorial) then *bad*

default is *good*

The normalized DIALOGUE QUALITY features result in a similar improvement in performance(65.9%).6 However, unlike the efficiency and experimental parameters features, the normalization of the dialogue quality features by dialogue length means that rules learned on the basis of these features are more likely to generalize. Adding the efficiency and normalized quality feature sets together (EFFICIENCY + NORMALIZED QUALITY) results in a significant performance improvement (69.7%) over EFFICIENCY alone. Figure

5 shows that this results in a classifier with three rules: one based on quality alone (percentage

of cancellations), one based on efficiency 6 The normalized versions of the quality features did better than the raw versions. alone (elapsed time), and one that consists of a Boolean combination of efficiency and quality features (elapsed time and percentage of rejections). The learned ruleset says that if the percentage of cancellations is greater than 6%, classify the dialogue as *bad*; if the elapsed time is greater than 282 seconds, and the

percentage of rejections is greater than 6%, classify it as *bad*; if the elapsed time is less than 90 seconds, classify it as *bad* 7 ; otherwise classify it as *good*. When multiple rules are applicable, RIPPER resolves any potential conflict by using the class that comes first in the ordering; when no rules are applicable, the default is used.

Figure 5: EFFICIENCY + NORMALIZED QUALITY

if (cancel% > =6) then *bad*

if (elapsed time > =282 secs)A(rejection% > 6) then *bad*

if (elapsed time < 90 secs) then *bad*

default is *good*

We discussed our acoustic *REJECTION%* results above, based on using the rejection thresholds that each system was actually run with. However, a posthoc analysis of our experimental data showed that our systems could have rejected substantially more misrecognitions with a rejection threshold that was lower than the thresholds picked by the sys-tem designers. (Of course, changing the thresh-olds in this way would have also increased the number of rejections of *correct* ASR outputs.) Re-call that the *PMISRECS%* experiments explored the use of different thresholds to predict misrecognitions. The best of these acoustic thresholds was *PMISRECS%3*, with accuracy 72.6%. This classifier learned that if the predicted percentage of misrecognitions using the threshold for that feature was greater than 8%, then the dialogue was predicted to be bad, otherwise it was good. This classifier per-forms significantly better than the BASELINE, *RE-JECTION%*

and EFFICIENCY classifiers. Similarly, *MEAN CONFIDENCE* is another acoustic feature, which averages confidence scores over all the non-rejected utterances in a dialogue. Since this feature is not tuned to the applications, we did not expect it to perform as well as the best *PMISRECS%* feature. However, the accuracy rate 7 This rule indicates dialogues too short for the user to have completed the task. Note that this rule could not be applied to adapting the system's behavior during the course of the dialogue. for the *MEAN CONFIDENCE* classifier (68.4%) is not statistically different than that for the *PMIS-RECS% 3* classifier. Furthermore, since the feature does not rely on picking an optimal threshold, it could be expected to better generalize to new dialogue situations. The classifier trained on (noisy) ASR lexical out-put *(ASR TEXT)* has access only to the speech recognizer's interpretation of the user's utterances. The *ASR TEXT* classifier achieves 72% accuracy, which is significantly better than the BASELINE, *REJEC-TION%* and EFFICIENCY classifiers. Figure 6 shows the rules learned from the lexical feature alone. The rules include lexical items that clearly indicate that a user is having trouble e.g. *help* and *cancel*. They also include lexical items that identify particular tasks for particular systems, e.g. the lexical item *p-m* identifies a task in TOOT.

Figure 6: *ASR TEXT* rules.

if (ASR text contains cancel) then *bad*

if (ASR text contains the) ^(ASR text contains get)(ASR textcontains TIMEOUT) then *bad*

if (ASR text contains today)^(ASR text contains on) then *bad*

if (ASR text contains the)^(ASR text contains p-m) then *bad*

if (ASR text contains to) then *bad*

if (ASR text contains help)^(ASR text contains the)^(ASR text contains read) then *bad*

if (ASR text contains help)^(ASR text contains previous) then *bad*

if (ASR text contains about) then *bad*

if (ASR text contains change-strategy) then *bad*

default is *good*

Note that the performance of many of the classifiers is statistically indistinguishable, e.g. the performance of the *ASR TEXT* classifier is virtually identical to the classifier *PMISRECS%3* and the EF-FICIENCY + QUALITY + EXP-PARAMS classifier. The similarity between the accuracies for a range of classifiers suggests that the information provided by different feature sets is redundant. As discussed above, each system and experimental condition resulted in dialogues that contained lexical items that were unique to it, making it possible to identify experimental conditions from the lexical items alone. Figure 7 shows the rules that RIPPER learned when it had access to all the features except for the lexical and acoustic features. In this case, RIPPER learns some rules that are specific to the TOOT system. Finally, the last row of Figure 4 suggests that a classifier that has access to ALL FEATURES may do better (77.4% accuracy) than those classifiers that

Figure 8: EFFICIENCY + QUALITY + EXP-PARAMS rules.

if (cancel% > 4)^(system = toot) then *bad*

if (system turns > 26)^(rejection% > 5) then *bad*

if (condition = mixed)^(user turns > 12) then *bad*

if (system = toot)^(user turns > 14) then *bad*

if (cancels > 1)^(timeout% > 11) then *bad*

if (elapsed time < 87 secs) then *bad*

default is *good*

This supports the conclusion that different feature sets provide redundant information, and could be substituted for each other to achieve the same performance. However, the ALL FEATURES classifier does perform significantly better than the EXP-PARAMS, DIALOGUE QUALITY (NORMALIZED), and *MEAN CONFIDENCE* classifiers. Figure 8 shows the decision rules that the ALL FEATURES classifier learns. Interestingly, this classifier does not find the features based on experimental parameters to be good predictors when it has other features to choose from. Rather it combines features representing acoustic, efficiency, dialogue quality and lexical information.

Figure 9: ALL FEATURES rules

if (mean confidence < -2.2)^(pmisrecs%4 > 6) then *bad*

if (pmisrecs%3 > 7)^(ASR text contains yes)^(mean confidence

< -1.9) then *bad*

if (cancel% > 4) then *bad*

if (system turns > 29)^(ASR text contains message) then *bad*

if (elapsed time < 90) then *bad*

default is *good*

4 Discussion

The experiments presented here establish several findings. First, it is possible to give an objective definition for poor speech recognition at the dialogue level, and to apply machine learning to build classifiers detecting poor recognition solely from features of the system log. Second, with appropriate sets of features, these classifiers significantly outperform the baseline percentage of the majority class. Third, the comparable performance of classifiers constructed from rather different feature sets (such as acoustic and lexical features) suggest that there is some redundancy between these feature sets (at least with respect to the task). Fourth, the fact that the best estimated accuracy was achieved using all of the features suggests that even problems that seem inherently acoustic may best be solved by exploiting higher-level information. This work differs from previous work in focusing on behavior at the (sub)dialogue level, rather than on identifying single misrecognitions at the utterance level (Smith, 1998; Levow, 1998; van Zanten,1998). The rationale is that a single misrecognition may not warrant a global change in dialogue strategy, whereas a user's repeated problems communicating with the system might warrant such a change. While we are not aware of any other work that has applied machine learning to detecting patterns suggesting that the user is having problems over the course of a dialogue, (Levow, 1998) has applied machine learning to identifying single misrecognitions. We are currently extending our feature set to include acoustic-prosodic features such as those used by Levow, in order to predict misrecognitions at both the dialogue level as well as the utterance level. We are also interested in the

extension and generalization of our findings in a number of additional directions. In other experiments, we demonstrated the utility of allowing the user to dynamically adapt the system's dialogue strategy at any point(s) during a dialogue. Our results show that dynamic adaptation clearly improves system performance, with the level of improvement sometimes a function of the system's initial dialogue strategy (Litman and Pan, 1999). Our next step is to incorporate classifiers such as those presented in this paper into a system in order to support dynamic adaptation according to recognition performance. Another area for future work would be to explore the utility of using alter-native methods for classifying dialogues as good or bad. For example, the user satisfaction measures we collected in a series of experiments using the PAR-ADISE evaluation framework (Walker et al., 1998c) could serve as the basis for such an alternative classification scheme. More generally, in the same way that learning methods have found widespread use in speech processing and other fields where large corporaare available, we believe that the construction and analysis of spoken dialogue systems is a ripe domain for machine learning applications.

5 Acknowledgements

Thanks to J. Chu-Carroll, W. Cohen, C. Kamm, M. Kan, R. Schapire, Y. Singer, B. Srinivas, and S. Whittaker for help with this research and/or paper.

References

Paul R. Cohen. 1995. *Empirical Methods for Arti-ficial Intelligence*. MIT Press, Boston.

William Cohen. 1996. Learning trees and rules with set-valued features. In *14th Conference of the American Association of Artificial Intelli-gence, AAAI*.

C. Kamm, S. Narayanan, D. Dutton, and R. Rite-nour. 1997. Evaluating spoken dialog systems for telecommunication services. In *5th European Conference on Speech Technology and Commu-nication, EUROSPEECH 97*.

Diane J. Litman, Shimei Pan, and Marilyn A. Walker. 1998. Evaluating Response Strategies in a Web-Based Spoken Dialogue Agent. In *Pro-ceedings of ACL/COLING 98: 36th Annual Meet-ing of the Association of Computational Linguis-tics*, pages 780– 787.

Elizabeth Shriberg, Elizabeth Wade, and Patti Price. 1992. Human-machine problem solving using spoken language systems (SLS): Factors affect-ing performance and user satisfaction. In *Pro-ceedings of the DARPA Speech and NL Workshop*, pages 49– 54.

Ronnie W. Smith. 1998. An evaluation of strate-gies for selectively verifying utterance meanings in spoken natural language dialog. *International Journal of Human-Computer Studies*, 48:627–647.

G. Veldhuijzen van Zanten. 1998. Adaptive mixed-initiative dialogue management. Technical Re-port 52, IPO, Center for Research on User-System Interaction.

Marilyn Walker, Donald Hindle, Jeanne Fromer, Giuseppe Di Fabbrizio, and Craig Mestel. 1997. Evaluating competing agent strategies for a voice email agent. In *Proceedings of the European Conference on Speech Communication and Tech-nology, EUROSPEECH97*.

M. Walker, J. Fromer, G. Di Fabbrizio, C. Mestel, and D. Hindle. 1998a. What can I say: Evaluat-ing a spoken language interface to email. In *Pro-ceedings of the Conference on Computer Human Interaction (CHI 98)*.

Marilyn A. Walker, Jeanne C. Fromer, and Shrikanth Narayanan. 1998b. Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *Proceedings of the 36th Annual Meeting of the Association of Com-putational Linguistics, COLING/ACL 98*, pages 1345– 1352.

Marilyn. A. Walker, Diane J. Litman, Candace. A. Kamm, and Alicia Abella. 1998c. Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech and Language*, 12(3).

Ilija Zeljkovic. 1996. Decoding optimal state se-quences with smooth state likelihoods. In *Inter-national Conference on Acoustics, Speech, and Signal Processing, ICASSP 96*, pages 129– 132.

Characterizing and Recognizing Spoken Corrections in Human-Computer Dialogue

Gina-Anne Levow
MIT AI Laboratory
Room 769, 545 Technology Sq
Cambridge, MA 02139
gina@ai.mit.edu

Abstract

Miscommunication in speech recognition systems is unavoidable, but a detailed characterization of user corrections will enable speech systems to identify when a correction is taking place and to more accurately recognize the content of correction utterances. In this paper we investigate the adaptations of users when they encounter recognition errors in interactions with a voice-in/voice-out spoken language system. In analyzing more than 300 pairs of original and repeat correction utterances, matched on speaker and lexical content, we found overall increases in both utterance and pause duration from original to correction. Interestingly, corrections of misrecognition errors (CME) exhibited significantly heightened pitch variability, while corrections of rejection errors (CRE) showed only a small but significant decrease in pitch minimum. CME's demonstrated much greater increases in measures of duration and pitch variability than CRE's. These contrasts allow the development of decision trees which distinguish CME's from CRE's and from original inputs at 70-75% accuracy based on duration, pitch, and amplitude features.

1 Introduction

The frequent recognition errors which plague speech recognition systems present a significant barrier to widespread acceptance of this technology. The difficulty of correcting system misrecognitions is directly correlated with user assessments of system quality. The increased probability of recognition errors immediately after an error compounds this problem. Thus, it becomes crucially important to characterize the differences between original utterances and user corrections of system recognition failures both in order to recognize when a user attempts a correction, indicating a prior recognition error, and to improve recognition accuracy on these problematic utterances. Analysis of data drawn from a field trial of a telephone-based voice-in/voice-out conversational system demonstrates significant differences between original inputs and corrections in measures of duration, pause, and pitch. These differences in turn aid in the development of decision trees which distinguish between new input and user corrections.

2 Overview

We begin by describing the framework in which the utterances used for these experiments were collected. We then describe the acoustic features used in the data analysis and decision tree design and motivate this choice by considering some related work. A report of the results of the descriptive analysis follows and demonstrates the significant differences among original inputs and different types of corrections. We then turn to the issues involved in the actual construction and assessment of decision trees which identify spoken corrections. The best of these classifiers achieves 75% success rates. Finally, we conclude with a plan to use these results to improve recognition accuracy on spoken corrections and ease recovery from human-computer miscommunication.

3 System, Subjects, and Errors

The utterances used in the design and analysis of the decision tree classifiers were drawn from approximately 60 hours of user interactions in a field trial of the Sun Microsystems SpeechActs system (Yankelovich et al., 1995). SpeechActs provides a voice-only interface to common desktop applications such as e-mail, calendar, weather reports, stock quotes, and

time and currency conversions. The system allows conversational interactions with speech recognition provided by BBN's HARK recognizer and speech synthesis through Centigram's TruVoice system. A sample interaction appears below.

SpeechActs:	You have 14 new messages and 47 old messages. The first new message is from Eric Baatz regarding "report completed."
User:	Let me hear it.
SpeechActs:	"The first draft is ready for your comments. Eric."
User:	Reply and include current message.
SpeechActs:	Preparing message to Eric Baatz. Begin recording after the tone. When finished, pause several seconds.
User:	Eric, I'll get those to you this afternoon.
SpeechActs:	Say cancel, send, or review.
User:	Send.
SpeechActs:	Message sent. What now?
User:	Next message.
SpeechActs:	New message two from Stuart Adams, subject "Can we meet at 3:00 today?"
User:	Switch to calendar... [1]

The field trial involved a group of nineteen subjects. Four of the participants were members of the system development staff, fourteen were volunteers drawn from Sun Microsystems' staff, and a final class of subjects consisted of one-time guest users There were three female and sixteen male subjects.

All interactions with the system were recorded and digitized in standard telephone audio quality format at 8kHz sampling in 8-bit mu-law encoding during the conversation. In addition, speech recognition results, parser results, and synthesized responses were logged. A paid assistant then produced a correct verbatim transcript of all user utterances and, by comparing the transcription to the recognition results, labeled each utterance with one of four accuracy codes as described below.

OK: recognition correct; action correct
Error Minor: recognition not exact; action correct
Error: recognition incorrect; action incorrect
Rejection: no recognition result; no action

Overall there were 7752 user utterances recorded, of which 1961 resulted in a label of either 'Error' or 'Rejection', giving an error rate of 25%. 1250 utterances, almost two-thirds of the errors, produced outright rejections, while 706 errors were substitution misrecognitions. The remainder of the errors were due to system crashes or parser errors. The probability of experiencing a recognition failure after a correct recognition was 16%, but immediately after an incorrect recognition it was 44%, 2.75 times greater. This increase in error likelihood suggests a change in speaking style which diverges from the recognizer's model. The remainder of this paper will identify common acoustic changes which characterize this error correction speaking style. This description leads to the development of a decision tree classifier which can label utterances as corrections or original input.

4 Related Work

Since full voice-in/voice-out spoken language systems have only recently been developed, little work has been done on error correction dialogs in this context. Two areas of related research that have been investigated are the identification of self-repairs and disfluencies, where the speaker self-interrupts to change an utterance in progress, and some preliminary efforts in the study of corrections in speech input.

In analyzing and identifying self-repairs, (Bear et al., 1992) and (Heeman and Allen, 1994) found that the most effective methods relied on identifying shared textual regions between the reparandum and the repair. However, these techniques are limited to those instances where a reliable recognition string is available; in general, that is not the case for most speech recognition systems currently available. Alternative approaches described in (Nakatani and Hirschberg, 1994) and (Shriberg et al., 1997), have emphasized acoustic-prosodic cues, including duration, pitch, and amplitude as discriminating features.

The few studies that have focussed on spoken corrections of computer misrecognitions, (Oviatt et al., 1996) and (Swerts and Ostendorf, 1995), also found significant effects of duration, and in Oviatt et al., pause insertion and length-

ening played a role. However, in only one of these studies was input "conversational", the other was a form-filling application, and neither involved spoken system responses, relying instead on visual displays for feedback, with potential impact on speaking style.

5 Error Data, Features, and Examples

For these experiments, we selected pairs of utterances: the first (original) utterance is the first attempt by the user to enter an input or a query; the second (repeat) follows a system recognition error, either misrecognition or rejection, and tries to correct the mistake in the same words as the original. For example,

SYSTEM SAID: Please say mail, calendar, weather, stock quotes or start over to begin again.
USER SAID:MAIL
SYSTEM HEARD: MAIL
CODE: OK
SYSTEM SAID: Switching to mail. Your first message is ..
USER SAID:Read message four eight nine
SYSTEM HEARD: "nothing"
CODE: Rejection
SYSTEM SAID: Sorry ?
USER SAID:Read message four eight nine
SYSTEM HEARD: "nothing"
CODE: Rejection
SYSTEM SAID: Still no luck. Speak clearly, but don't over emphasize .
USER SAID:Go to message four eight nine
SYSTEM HEARD: Go to message four please umm
CODE: ERROR
SYSTEM SAID: I heard goto new message 4.

In total, there were 302 of these original-repeat pairs: 214 resulting from rejections, and 88 from misrecognitions.

Following (Oviatt et al., 1996), (Shriberg et al., 1997), and (Ostendorf et al., 1996), we coded a set of acoustic-prosodic features to describe the utterances. These features fall into four main groups: durational, pause, pitch, and amplitude. We further selected variants of these feature classes that could be scored automatically, or at least mostly automatically with some minor hand-adjustment. We hoped that these features would be available during the recognition process so that ultimately the original-repeat correction contrasts would be identified automatically.

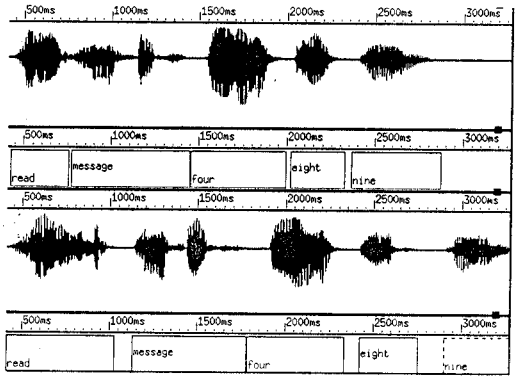


Figure 1: A lexically matched pair where the repeat (bottom) has an 18% increase in total duration and a 400% increase in pause duration.

5.1 Duration

The basic duration measure is total utterance duration. This value is obtained through a two-step procedure. First we perform an automatic forced alignment of the utterance to the verbatim transcription text using the OGI CSLU CSLUsh Toolkit (Colton, 1995). Then the alignment is inspected and, if necessary, adjusted by hand to correct for any errors, such as those caused by extraneous background noise or non-speech sounds. A typical alignment appears in Figure 1. In addition to the simple measure of total duration in milliseconds, a number of derived measures also prove useful. Some examples of such measures are speaking rate in terms of syllables per second and a ratio of the actual utterance duration to the mean duration for that type of utterance.

5.2 Pause

A pause is any region of silence internal to an utterance and longer than 10 milliseconds in duration. Silences preceding unvoiced stops and affricates were not coded as pauses due to the difficulty of identifying the onset of consonants of these classes. Pause-based features include number of pauses, average pause duration, total pause duration, and silence as a percentage of total utterance duration. An example of pause

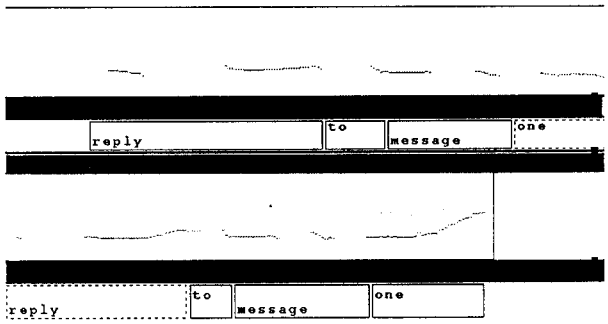


Figure 2: Contrasting Falling (top) and Rising (bottom) Pitch Contours

insertion and lengthening appear in Figure 1.

5.3 Pitch

To derive pitch features, we first apply the F0 (fundamental frequency) analysis function from the Entropic ESPS Waves+ system (Secrest and Doddington, 1993) to produce a basic pitch track. Most of the related work reported above had found relationships between the magnitude of pitch features and discourse function rather than presence of accent type, used more heavily by (Pierrehumbert and Hirschberg, 1990), (Hirschberg and Litman, 1993). Thus, we chose to concentrate on pitch features of the former type. A trained analyst examines the pitch track to remove any points of doubling or halving due to pitch tracker error, non-speech sounds, and excessive glottalization of ≥ 5 sample points. We compute several derived measures using simple algorithms to obtain F0 maximum, F0 minimum, F0 range, final F0 contour, slope of maximum pitch rise, slope of maximum pitch fall, and sum of the slopes of the steepest rise and fall. Figure 2 depicts a basic pitch contour.

5.4 Amplitude

Amplitude, measuring the loudness of an utterance, is also computed using the ESPS Waves+ system. Mean amplitudes are computed over all voiced regions with amplitude $\geq$ 30dB. Amplitude features include utterance mean amplitude, mean amplitude of last voiced region, amplitude of loudest region, standard deviation, and difference from mean to last and maximum to last.

6 Descriptive Acoustic Analysis

Using the features described above, we performed some initial simple statistical analyses to identify those features which would be most useful in distinguishing original inputs from repeat corrections, and corrections of rejection errors (CRE) from corrections of misrecognition errors (CME). The results for the most interesting features, duration, pause, and pitch, are described below.

6.1 Duration

Total utterance duration is significantly greater for corrections than for original inputs. In addition, increases in correction duration relative to mean duration for the utterance prove significantly greater for CME's than for CRE's.

6.2 Pause

Similarly to utterance duration, total pause length increases from original to repeat. For original-repeat pairs where at least one pause appears, paired t-test on log-transformed data reveal significantly greater pause durations for corrections than for original inputs.

6.3 Pitch

While no overall trends reached significance for pitch measures, CRE's and CME's, when considered separately, did reveal some interesting contrasts between corrections and original inputs within each subset and between the two types of corrections. Specifically, male speakers showed a small but significant decrease in pitch minimum for CRE's.

CME's produced two unexpected results. First they displayed a large and significant increase in pitch variability from original to repeat as measured the slope of the steepest rise, while CRE's exhibited a corresponding decrease rising slopes. In addition, they also showed significant increases in steepest rise measures when compared with CRE's.

7 Discussion

The acoustic-prosodic measures we have examined indicate substantial differences not only between original inputs and repeat corrections, but also between the two correction classes, those in response to rejections and those in response to misrecognitions. Let us consider the relation of these results to those of related work

and produce a more clear overall picture of spoken correction behavior in human-computer dialogue.

7.1 Duration and Pause: Conversational to Clear Speech

Durational measures, particularly increases in duration, appear as a common phenomenon among several analyses of speaking style [(Oviatt et al., 1996), (Ostendorf et al., 1996), (Shriberg et al., 1997)]. Similarly, increases in number and duration of silence regions are associated with disfluencies (Shriberg et al., 1997), self-repairs (Nakatani and Hirschberg, 1994), and more careful speech (Ostendorf et al., 1996) as well as with spoken corrections (Oviatt et al., 1996). These changes in our correction data fit smoothly into an analysis of error corrections as invoking shifts from conversational to more "clear" or "careful" speaking styles. Thus, we observe a parallel between the changes in duration and pause from original to repeat correction, described as conversational to clear in (Oviatt et al., 1996), and from casual conversation to carefully read speech in (Ostendorf et al., 1996).

7.2 Pitch

Pitch, on the other hand, does not fit smoothly into this picture of corrections taking on clear speech characteristics similar to those found in carefully read speech. First of all, (Ostendorf et al., 1996) did not find any pitch measures to be useful in distinguishing speaking mode on the continuum from a rapid conversational style to a carefully read style. Second, pitch features seem to play little role in corrections of rejections. Only a small decrease in pitch minimum was found, and this difference can easily be explained by the combination of two simple trends. First, there was a decrease in the number of final rising contours, and second, there were increases in utterance length, that, even under constant rates of declination, will yield lower pitch minima. Third, this feature produces a divergence in behavior of CME's from CRE's.

While CRE's exhibited only the change in pitch minimum described above, corrections of misrecognition errors displayed some dramatic changes in pitch behavior. Since we observed that simple measures of pitch maximum, minimum, and range failed to capture even the basic contrast of rising versus falling contour, we extended our feature set with measures of slope of rise and slope of fall. These measures may be viewed both as an attempt to create a simplified form of Taylor's rise-fall-continuation model (Taylor, 1995) and as an attempt to provide quantitative measures of pitch accent. Measures of pitch accent and contour had shown some utility in identifying certain discourse relations [(Pierrehumbert and Hirschberg, 1990), (Hirschberg and Litman, 1993). Although changes in pitch maxima and minima were not significant in themselves, the increases in rise slopes for CME's in contrast to flattening of rise slopes in CRE's combined to form a highly significant measure. While not defining a specific overall contour as in (Taylor, 1995), this trend clearly indicates increased pitch accentuation. Future work will seek to describe not only the magnitude, but also the form of these pitch accents and their relation to those outlined in (Pierrehumbert and Hirschberg, 1990).

7.3 Summary

It is clear that many of the adaptations associated with error corrections can be attributed to a general shift from conversational to clear speech articulation. However, while this model may adequately describe corrections of rejection errors, corrections of misrecognition errors obviously incorporate additional pitch accent features to indicate their discourse function. These contrasts will be shown to ease the identification of these utterances as corrections and to highlight their contrastive intent.

8 Decision Tree Experiments

The next step was to develop predictive classifiers of original vs repeat corrections and CME's vs CRE's informed by the descriptive analysis above. We chose to implement these classifiers with decision trees (using Quinlan's (Quinlan, 1992) C4.5) trained on a subset of the original-repeat pair data. Decision trees have two features which make them desirable for this task. First, since they can ignore irrelevant attributes, they will not be misled by meaningless noise in one or more of the 38 duration, pause, pitch, and amplitude features coded. Since these features are probably not all important, it is desir-

able to use a technique which can identify those which are most relevant. Second, decision trees are highly intelligible; simple inspection of trees can identify which rules use which attributes to arrive at a classification, unlike more opaque machine learning techniques such as neural nets.

8.1 Decision Trees: Results & Discussion

The first set of decision tree trials attempted to classify original and repeat correction utterances, for both correction types. We used a set of 38 attributes: 18 based on duration and pause measures, 6 on amplitude, five on pitch height and range, and 13 on pitch contour. Trials were made with each of the possible subsets of these four feature classes on over 600 instances with seven-way cross-validation. The best results, 33% error, were obtained using attributes from all sets. Duration measures were most important, providing an improvement of at least 10% in accuracy over all trees without duration features.

The next set of trials dealt with the two error correction classes separately. One focussed on distinguishing CME's from CRE's, while the other concentrated on differentiating CME's alone from original inputs. The test attributes and trial structure were the same as above. The best error rate for the CME vs. CRE classifier was 30.7%, again achieved with attributes from all classes, but depending most heavily on durational features. Finally the most successful decision trees were those separating original inputs from CME's. These trees obtained an accuracy rate of 75% (25% error) using similar attributes to the previous trials. The most important splits were based on pitch slope and durational features. An exemplar of this type of decision tree in shown below.

```
normduration1 > 0.2335 : r (39.0/4.9)
normduration1 <= 0.2335 :
|normduration2 <= 20.471 :
||normduration3 <= 1.0116 :
|||normduration1 > -0.0023 : o (51/3)
|||normduration1 <= -0.0023 :
|||| pitchslope > 0.265 : o (19/4))
|||| pitchslope <= 0.265 :
||||| pitchlastmin <= 25.2214:r(11/2)
||||| pitchlastmin > 25.2214:
|||||| minslope <= -0.221:r(18/5)
|||||| minslope > -0.221:o(15/5)
||normduration3 > 1.0116 :
|||normduration4 > 0.0615 : r (7.0/1.3)
|||normduration4 <= 0.0615 :
||||normduration3 <= 1.0277 : r (8.0/3.5)
||||normduration3 > 1.0277 : o (19.0/8.0)
|normduration2 > 20.471 :
|| pitchslope <= 0.281 : r (24.0/3.7)
|| pitchslope > 0.281 : o (7.0/2.4)
```

These decision tree results in conjunction with the earlier descriptive analysis provide evidence of strong contrasts between original inputs and repeat corrections, as well as between the two classes of corrections. They suggest that different error rates after correct and after erroneous recognitions are due to a change in speaking style that we have begun to model.

In addition, the results on corrections of misrecognition errors are particularly encouraging. In current systems, all recognition results are treated as new input unless a rejection occurs. User corrections of system misrecognitions can currently only be identified by complex reasoning requiring an accurate transcription. In contrast, the method described here provides a way to use acoustic features such as duration, pause, and pitch variability to identify these particularly challenging error corrections without strict dependence on a perfect textual transcription of the input and with relatively little computational effort.

9 Conclusions & Future Work

Using acoustic-prosodic features such as duration, pause, and pitch variability to identify error corrections in spoken dialog systems shows promise for resolving this knotty problem. We further plan to explore the use of more accurate characterization of the contrasts between original and correction inputs to adapt standard recognition procedures to improve recognition accuracy in error correction interactions. Helping to identify and successfully recognize spoken corrections will improve the ease of recovering from human-computer miscommunication and will lower this hurdle to widespread acceptance of spoken language systems.

References

J. Bear, J. Dowding, and E. Shriberg. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Proceedings of the ACL*, pages 56–63, University of Delaware, Newark, DE.

D. Colton. 1995. Course manual for CSE 553 speech recognition laboratory. Technical Report CSLU-007-95, Center for Spoken Language Understanding, Oregon Graduate Institute, July.

P.A. Heeman and J. Allen. 1994. Detecting and correcting speech repairs. In *Proceedings of the ACL*, pages 295–302, New Mexico State University, Las Cruces, NM.

Julia Hirschberg and Diane Litman. 1993. Empirical studies on the disambiguation of cue phrases. *Computational linguistics*, 19(3):501–530.

C.H. Nakatani and J. Hirschberg. 1994. A corpus-based study of repair cues in spontaneous speech. *Journal of the Acoustic Society of America*, 95(3):1603–1616.

M. Ostendorf, B. Byrne, M. Bacchiani, M. Finke, A. Gunawardana, K. Ross, S. Roweis, E. Shribergand D. Talkin, A. Waibel, B. Wheatley, and T. Zeppenfeld. 1996. Modeling systematic variations in pronunciation via a language-dependent hidden speaking mode. In *Proceedings of the International Conference on Spoken Language Processing.* supplementary paper. .

S.L. Oviatt, G. Levow, M. MacEarchern, and K. Kuhn. 1996. Modeling hyperarticulate speech during human-computer error resolution. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 801–804.

Janet Pierrehumbert and Julia Hirschberg. 1990. The meaning of intonational contours in the interpretation of discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 271–312. MIT Press, Cambridge, MA.

J.R. Quinlan. 1992. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

B. G. Secrest and G. R. Doddington. 1993. An integrated pitch tracking algorithm for speech systems. In *ICASSP 1993.*

E. Shriberg, R. Bates, and A. Stolcke. 1997. A prosody-only decision-tree model for disfluency detection. In *Eurospeech '97.*

M. Swerts and M. Ostendorf. 1995. Discourse prosody in human-machine interactions. In *Proceedings of the ECSA Tutorial and Research Workshop on Spoken Dialog Systems - Theories and Applications.*

Paul Taylor. 1995. The rise/fall/continuation model of intonation. *Speech Communication*, 15:169–186.

N. Yankelovich, G. Levow, and M. Marx. 1995. Designing SpeechActs: Issues in speech user interfaces. In *CHI '95 Conference on Human Factors in Computing Systems*, Denver, CO, May.