

Abstract:

Music recording, like driving a car, requires the use of one's hands for other purposes, and is therefore a good application for voice commands. I implement and measure the performance of basic keyword voice commands such as "Play" and "Stop". Going beyond that, I expound upon the need to name recorded audio tracks, and to control parameters of those tracks, referring to them by name. These are commands such as "Mute the drums", or "Solo the piano", where "drums" and "piano" are the names of recorded audio tracks.

Experiments with several test subjects are conducted, and it was found that the system would respond reliably, although much less so initially for persons with heavy accents. For all users, the recognition performance improved over time.

Because the recognition accuracy for named track commands was found to be consistently lower than for keyword commands, I devised and implemented techniques to improve the recognition accuracy of assigning names and of using names in subsequent commands. It was found that once the correct name has been assigned to a track, the name can be loaded into the speech recognition engine as if it were a keyword. This results in similar recognition accuracy for named track commands as for keyword commands.

VOICE COMMANDS
TO CONTROL RECORDING SESSIONS

By

John Martin Goddard
B.S., B.A. University of Colorado at Boulder, 1989

Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science
in Computer Engineering in the Graduate School of Syracuse University

September 2011

Approved: _____

Professor James Fawcett

Date: _____

Copyright 2011 John Martin Goddard

All Rights Reserved

Contents

List of Figures	viii
List of Tables.....	x
Acknowledgments.....	xi
Chapter 1: Introduction	1
Description of the Problem Space.....	2
Description of the Music Recording Workflow	3
Recording Basic Tracks and Overdubs.....	6
Why Experiments Are Necessary	8
Objectives: Precise Statement of Intended Research	9
Areas of Research Performed.....	10
Statement of Expected Contributions	13
Relevant Research Sources	17
Survey of Prior Work on Voice Command Systems	21
Brief Discussion of Speech Recognition Technology:	27
Research Questions	29
Chapter 2: Controlling The Recording Process By Voice	33
Defining the Problem.....	34
Experimental Test Setup	36
Windows Speech Recognition (WSR) Issues.....	37
User Interface Considerations:	39
Factors Affecting Recognition Accuracy:.....	47
Potential Problems.....	49
Description of Experimental Apparatus Software Design:	52
SayPlay	52
Audacity	52
Overview of Experimental Procedure	53

Recording Success and Failure of Command Events in a Log File:	53
Voice Command Failure Modes	55
Test Plan Overview	57
Chapter 3: Command Recognition Accuracy	59
Basic Commands:.....	60
Keyword Commands:	61
Improvement in Command Accuracy	62
Chapter 4: Naming and Referencing Tracks by Name	68
Flexible Grammar Constructions	68
Overview of Improving Name Command Recognition	70
Improving Accuracy Assigning Names	72
Elaboration:.....	72
Spelling it Out.....	74
Improving Accuracy in Referring to Assigned Names	75
Adding Names into the Windows Speech Recognition Dictionary.....	75
Loading Names as “Choices” in a grammar in the Recognition Engine ..	77
Discussion of results:	79
Discussion of results of Tricky Names and Dictionary/Grammar.....	83
Elaboration, and Loading Grammars using Homophone Pairs	84
Discussion of results:	89
Discussion of anomalies:	91
Conclusions of Homophone Experiments:.....	93
Chapter 5: Contribution Summary And Future Work	95
Future Work on SayPlay.....	98
Possible Future Work:.....	100
Conclusion.....	106
Appendix A: Storyboard Descriptions of Recording Session Workflow	108

Recording Music:	114
Recording Voice:.....	115
Appendix B: The Demonstration System	116
Appendix C: Setting Up Windows Speech Recognition.....	124
Appendix D: The Windows Speech Dictionary.....	132
Prevent a Word from Being Recognized.....	134
Appendix E: Test Procedure.....	136
Appendix F: Summary of Rules Applied to Wildcard Text Strings	150
Appendix G: Synopsis of Experiments	151
Baseline Performance.....	152
Keyword Commands	153
Field Tests with Multiple Users.....	154
Add Tricky Names to Dictionary	155
Successive Improvements:	156
Load Therein into Grammar.....	156
Add Tricky Names to Dictionary, then Load them into Grammar.....	157
Elaboration using Homophones.....	158
Homophones to Show Effectiveness of Loading Names into Grammar	159
Appendix H: Bibliography.....	160
Appendix I: Glossary.....	175

LIST OF FIGURES

Figure 1 Recording and editing tracks in Audacity	3
Figure 2 Recording workflow diagram.....	5
Figure 3 Recording Overdubs workflow diagram.....	7
Figure 4 Audacity waveform display of audio tracks.....	35
Figure 5 Recording equipment used in experiments	36
Figure 6 Activity diagram for assigning “Banjo” to the recorded track.....	44
Figure 7 Activity Diagram for loading track names into the loaded Grammar.....	46
Figure 8 SayPlay logfile excerpt	54
Figure 9 Baseline performance test, mean WSR Confidence	60
Figure 10 Mean Confidence value returned from WSR Engine	62
Figure 11 Cumulative successes for 5 test subjects.....	63
Figure 12 Errors by category of error.....	65
Figure 13 Success rates of Keywords Commands vs. Named Track Commands.....	67
Figure 14 Grammar structure to allow tracks to be named.	68
Figure 15 Grammar structure for issuing track commands by name.....	69
Figure 16 Cumulative Success Before/After Adding Name to Speech Dictionary.....	76
Figure 17 Recognition Rates after Successive Improvements	79
Figure 18 Recognition Rates Before/After Loading “Theremin” into Grammar.....	81
Figure 19 Cumulative Successes Before and After loading into loaded grammar	83
Figure 20 Recognition Accuracy when assigning names using Elaboration	86
Figure 21 Recognition Accuracy after loading homophones into loaded grammar.....	88
Figure 22 Grammar structure to rename an existing keyword command.....	99
Figure 23 Recording Overdubs.....	109
Figure 24 Workflow for recording Overdubs.....	111
Figure 25 Audacity User Interface with annotations	116
Figure 26 The SayPlay program written for this research.....	117
Figure 27 Class Diagram of the CommandFamily in SayPlay	119
Figure 28 Context Diagram of SayPlay and Audacity applications.....	120
Figure 29 Activity Diagram for naming a track by spelling out the name.....	121
Figure 30 Activity Diagram for naming a track using “Elaboration”.....	122
Figure 31 Activity Diagram for the Track Command to Play a Named Track	123
Figure 32 Speech Recognition microphone setup	125
Figure 33 Speech Recognition Microphone Setup for levels.....	126
Figure 34 Speech Recognition Control Panel Advanced Options.....	127
Figure 35 Speech Recognition profiles.....	128
Figure 36 Selecting the Speech Recognition Control Panel for Microphone Level.....	129
Figure 37 Speech Recognition Control Panel Options, initiating a training session	130
Figure 38 Speech Recognition Voice Training dialog	130
Figure 39 Adding a word to the Windows Speech Dictionary.	132

Figure 40 Record a pronunciation of new word added.	133
Figure 41 Prevent a Name from Being Recognized	134
Figure 42 Prevent a word from being dictated.	135
Figure 43 Audacity with custom SayPlay script status display	137
Figure 44 Windows Speech Recognition Profiles.	138

LIST OF TABLES

Table 1 Speech Recognition Research Sources (Authors) by Category	21
Table 2 .Net System.Speech.Recognition highlights.....	27
Table 3 Test Plan	58
Table 4 Confused and Tricky Names	82
Table 5 Success/Attempt ratios for Adding Tricky Names to Dictionary and Grammar...	82
Table 6 Terms referring to track commands.....	113
Table 7 Naming Tracks and Takes	115
Table 8 Test 1 Procedure: Baseline Measure	139
Table 9 Test 2 All Keyword Commands	142
Table 10 Test 3 Test Procedure: Field Trials	142
Table 11 Test 4 Test Procedure: Tricky Names	142
Table 12 Test 5: Effectiveness Adding Theremin to Dictionary and Grammar.....	143
Table 13 Test 6: Adding Tricky Names to Dictionary and Grammar	143
Table 14 Test 7: Homophone Tests of Elaboration and Loading into Grammar.	148
Table 15: List of Rules Applied to Names Strings When Assigning Them.	150
Table 16: Overview of Experiments and Results.....	151

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Professor Fawcett for his advice and guidance in the preparation of this manuscript.

The author also wishes to thank his Thesis Advisor, Dr. James Fawcett, and Dr. Douglas Quin for his patience and insight acting as a test subject. The author also wishes to thank the Thesis committee members, including Dr. Nancy McCracken and Dr. Fred Schlereth.

CHAPTER 1: INTRODUCTION

Research statement, expected contributions, and relevant research of others.

Hands-free computing by voice command has been successfully used by radiologists to take transcription of notes on X-Ray pictures [Ernst], to allow persons to remain productive working at a computer following disabilities such as computer related repetitive stress injuries [Hubbell]. It is also used by doctors in taking dictation [Caskey], control of home appliances [Chun-Liang] and drivers when interacting with automotive GPS navigation systems [Chang] and smart phones [Cohen] [Schuricht]. Hands-free computing can also be quite useful to streamline the process of recording music, since a musician's hands are frequently needed for musical performance.

But, if an instrumentalist such as a pianist or guitarist wants to record themselves using a laptop computer, they have to reach for the touch pad or mouse to click on icons of tape machine controls, or press keyboard shortcuts. If the hands-free computing paradigms used for other applications could be used for audio recording, then the musicians would not have to reach away from their instruments to start and stop a recording. A producer would not need a tape operator to record an ensemble.

Description of the Problem Space

Today, music recording is primarily performed using computer-based systems called Digital Audio Workstations (DAW). The DAW organizes recorded audio into tracks, just as it would be organized when recorded onto magnetic tape, and mixed within a mixing console with controls per track such as a level fader, left/right pan control, solo and mute buttons, and other controls. The picture below shows a subset of audio tracks with Mute and Solo controls indicated.

While the specific software shown is Audacity, the horizontal audio waveforms are a standard way to show the composition of audio tracks making up a recording session. Note only 4 audio tracks are currently visible in the display.

The fact that a scroll-bar is necessary for moving the tracks which are out of sight above and below into view, in order to operate the track controls, means several mouse moves are required just to mute or solo a track.

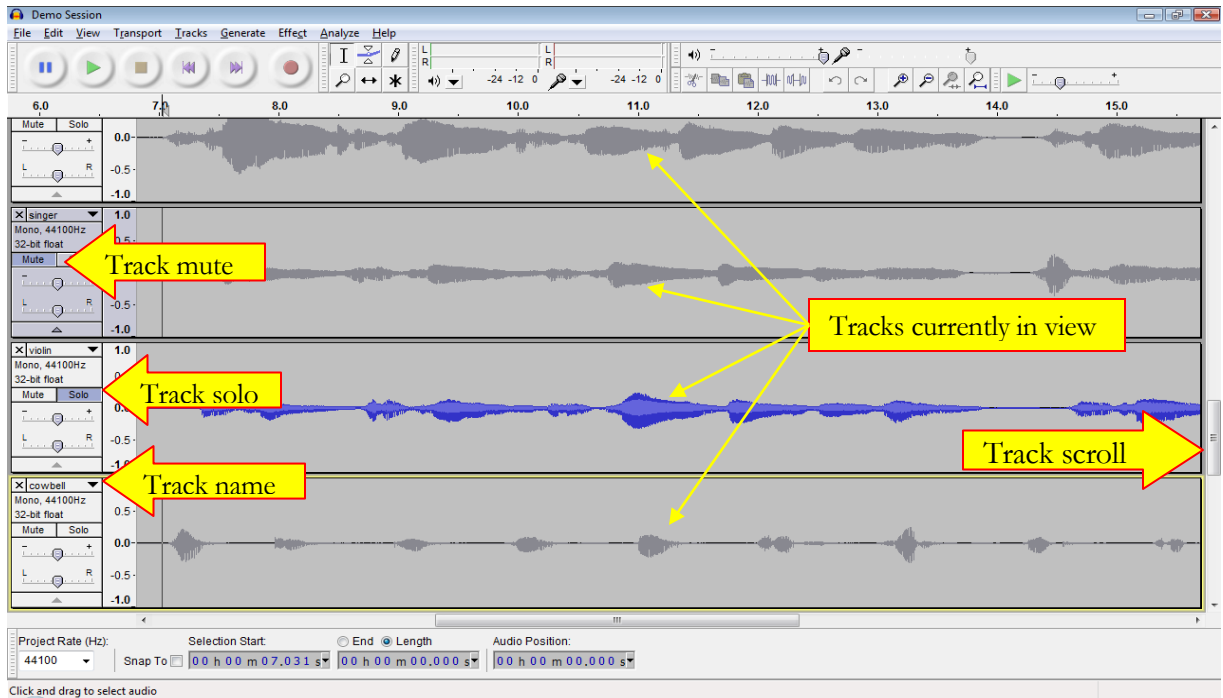


Figure 1 Recording and editing tracks in Audacity

Description of the Music Recording Workflow

The process of recording is basically the same as shown in Figure 2 Recording workflow diagram. The end-to-end workflow for recording music is broadly defined as follows:

Recording Basic Tracks: Record initial run-through of the piece, generally assigning one track per instrument

Recording Overdubs: Play back the “Basic Tracks”, while recording a new voice or instrument that is playing along with the basic tracks.

Editing: Cut, paste, shift, replace portions of a recorded track

Mixing: Blend and balance all recorded tracks for an optimal stereo listening experience.

Final Mix down: After all recording and editing of tracks is finished, make a final stereo mix.

Mastering: adjust final mix to the target media, whether it is for audio CD, DVD, MP3, movie soundtrack, etc.

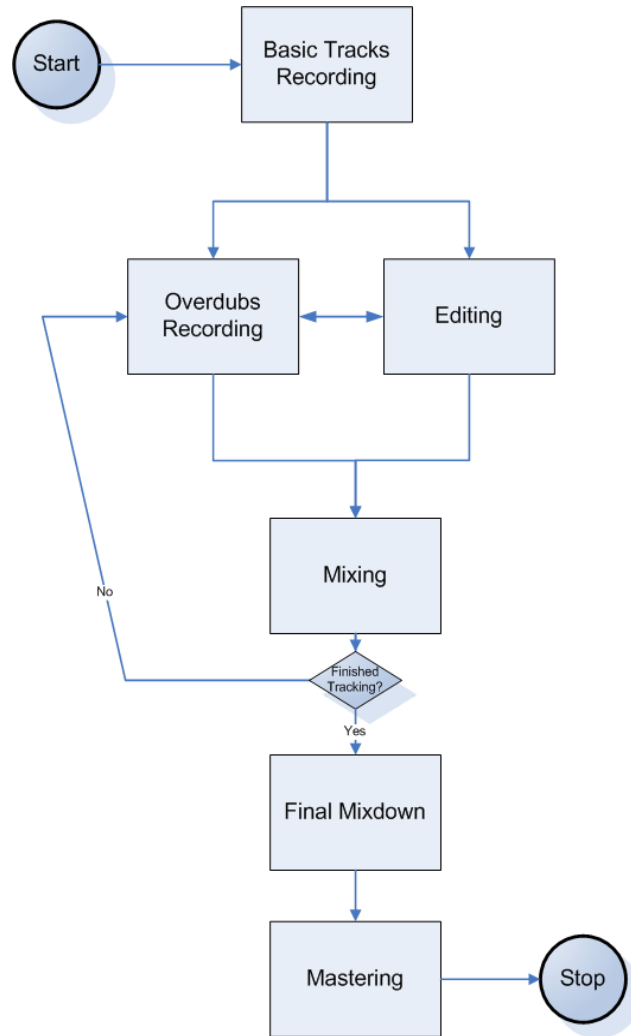


Figure 2 Recording workflow diagram

Recording Basic Tracks and Overdubs

Recording (both basic tracks and overdubs) involves the capture of a performance, while the other processes (editing, mixing, and mastering) are focused on the already-recorded audio. Therefore the recording process would benefit most by being hands-free, and therefore, this is the focus of this research. Figure 3 Recording Overdubs workflow diagram shows recording basic tracks and overdubs.

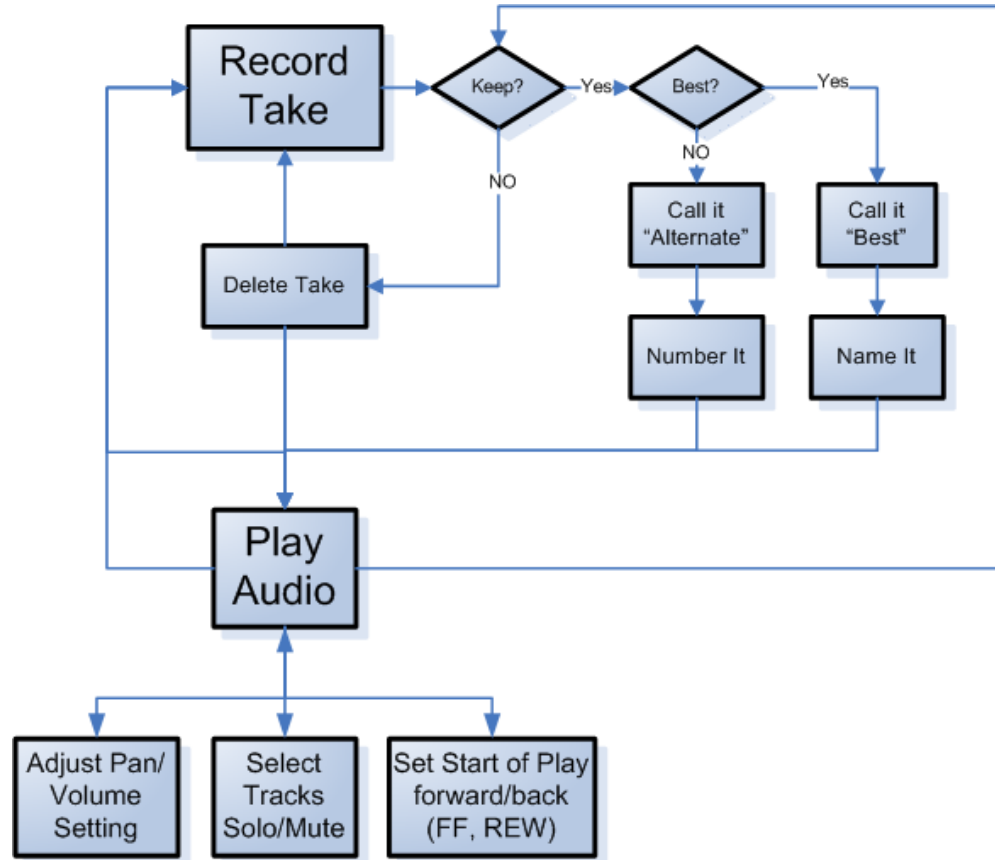


Figure 3 Recording Overdubs workflow diagram

Record Take: Begin recording, then perform music, voice over etc., then stop recording. Delete take if not salvageable.

Play Audio: Audition recorded tracks to listen for performance problems.

Solo: Selects track to hear alone, while others are temporarily turned off. Other tracks can also be heard together if their solo control is also activated.

Mute: Temporarily turn off this track

Pan: shift left or right in stereo mix

For examples of the kind of dialog that usually takes place during a recording session, please see Appendix A: Storyboard Descriptions of Recording Session Workflow.

Why Experiments Are Necessary

The complexity of the music recording workflow and the demands of a real-time recording session mean that an exercise of discovery is required in order to learn where the process works and where it breaks down.

Windows Speech Recognition was chosen over the other speech recognition engines because it is built into the Windows (NT, Vista and later) operating system, and because it would be easiest for developing custom speech application

software to work with Audacity, the music recording application. There are also extensive online documentation¹, a developer community², and source code³.

The unpredictable nature of Windows Speech Recognition requires characterization, and in order to make statistically significant claims, numerous experimental repetitions are necessary. The areas where speech recognition is most unreliable are the following: 1. before training, 2. after training by individuals with heavy accents, 3. incorrect name recognition, and 4. user error based on improperly phrased commands.

Objectives: Precise Statement of Intended Research

The use of speech recognition for command and control of a digital audio workstation is a natural fit because recording music requires that one's hands are free to play the instruments being recorded. Because recording sessions have traditionally been managed by a producer issuing commands by voice, both to the

¹ http://wiki.audacityteam.org/wiki/Audacity_Wiki_Home_Page

² <http://audacity.sourceforge.net/community/developers>

³ <http://audacity.sourceforge.net/download/>

recording engineer and to tape operators, it is natural to model those voice commands in a system that automates the tape operator's function. The goal of this research is to enhance the digital audio workstation with voice commands, and to study the experience to make it better than using the mouse and keyboard.

Areas of Research Performed

A "hands-free" recording tool was developed and instrumented to write to an event log the details of each speech recognition event. This allows for analysis of recognition accuracy and effectiveness of voice commands to control recording sessions.

Speech Recognition performance is measured as a ratio of successful commands to the total number of commands issued, for each experiment. Performance is measured for keyword commands (Play, Stop, Etc.) and for references to named objects, both assigning names and using the names for issuing commands.

Research Areas investigated for Improving Performance:

User defined names: The facility for the user to name tracks or other entities, so that subsequent commands could refer to those things by name, has been

developed for this research. In addition, we devised the following techniques to improve recognition accuracy of assigned names:

Elaboration: Allow user to elaborate on a name using the phrases: “As in...” or “Like...”, to resolve homophones, or other misrecognized names.

“As Follows”, allow the user to indicate that the following phrase is to be used as a name. This name could include “Like” or “As In”, and these words will not be omitted from the name, as they are when using them for Elaboration, as described above. Allow user to say “Quote” at the beginning and “Unquote” at end of long phrase to be used as a name. In the same way that “As Follows” is used to indicate inclusion of all of the following words into the name, “Quote” and “Unquote” are also used to capture the entire phrase enclosed by those key words.

Spell it out: Allow user to spell out a name. This is useful in cases where the name is so difficult to recognize that it seems impossible to assign any other way.

We investigated the following techniques and features to improve the performance of referring to tracks once their name has been assigned:

Load the names of items into the grammar structure⁴, so that names are recognized as keywords, instead of using Dictation Speech Recognition. The technique to do this automatically was developed and implemented for this research.

Add names to the Windows Speech Recognition Dictionary⁵

Prevent dictation of confused names in the Windows Speech Dictionary⁶.

We utilized the Microsoft published ways to improve speech recognition accuracy:

Setup the microphone. This step is required. It sets the signal level that is used during Speech Recognition⁷.

Perform training sessions⁸. Microsoft provides 2 training sessions, each of which may be repeated if desired or necessary. The user reads aloud the text provided

⁴ See: Figure 7 Activity Diagram for loading track names into the loaded Grammar

⁵ See: Figure 39 Adding a word to the Windows Speech Dictionary.

⁶ See Prevent a Word from Being Recognized in Appendix D: The Windows Speech Dictionary

⁷ See Appendix C: Setting Up Windows Speech Recognition

⁸ See Appendix C: Setting Up Windows Speech Recognition

and the Windows Speech Recognition (WSR) system adjusts internal parameters according to the pronunciation of particular words or word fragments (phonemes).

Note: Allow “Document Scanning” was not enabled. This allows the system to scan user’s documents for words and phrases. It is unknown how much this helps recognition accuracy.

It was determined that other user’s speech profiles benefit from the addition of a new word into the dictionary of another’s speech profile. An experiment was run, using 2 different speech profiles, to demonstrate that a new word becomes recognizable from a second profile, if added to the first user profile.

Statement of Expected Contributions

Contribution 1: A system which responds to voice commands and supports the multi-track audio recording workflow was developed and measured for accuracy. The prototype system demonstrates the usefulness and effectiveness of the proposed hands-free workflow, and allows measurements to be made by logging status for each speech command issued.

Contribution 2: Changes and improvements made to Audacity, an open source audio recorder and editor, to support track commands issued by name. These include the means of getting and setting the state of all tracks Mute or Solo settings. Also added to Audacity was the means for setting a track parameter, such as Mute, Solo, Gain or Pan, given the name of the track to which to apply the change.

Contribution 3: Analyzed experimental data gathered from several test subjects in using the system to perform basic recording tasks. Conclusions are drawn of the effect of accent on the rate of performance improvement provided by the Speech Recognition Engine's machine learning capability.

Contribution 4: Naming tracks and then referring to them by name is developed as a way to improve system flexibility and hands-free usability. Named tracks are easier to refer to than numbered tracks, because the user has assigned the names from a meaningful connection to the content within. For example, the name of the instrument or the person performing, or the part within the arrangement, such as banjo, Paul's Vocal, or rhythm section, respectively, can be used.

Contribution 5: The technique of loading all track names as choices into a grammar in the Speech Recognition Engine is explored⁹. This provides an improvement in performance of commands that refer to named tracks. When names are added to the grammar in this way, performance depends on keyword speech recognition, where the possible names are known in advance. These names are easier to recognize than having to depend on dictation speech recognition, which must accurately recognize any word not known in advance.

Contribution 6: I developed and analyzed the Elaboration technique to improve name recognition when assigning new names (not known in advance) using dictation recognition. Homophone pairs are used to test whether the Elaboration technique helps determine the correct sense of the intended homophone pair. This is like a person clarifying a statement such as: “Bass as in bass guitar, not base as in baseball”¹⁰.

⁹ See: Figure 7 Activity Diagram for loading track names into the loaded Grammar

¹⁰ See Figure 30 Activity Diagram for naming a track using “Elaboration”

Contribution 7: Two features provided by Windows Speech Recognition, namely adding a word to the Speech Dictionary¹¹, and preventing Recognition of confused words¹², are measured as a way of improving recognition accuracy of names. Even adding words already in the speech dictionary helps, because the word is spoken once when being added to the dictionary, providing the system with an audio example.

Measurement of the effectiveness of the system requires counting the number of failures in a given number of attempts. Measuring accuracy after implementing improvements shows the validity of improvements. Improved performance is measured using the statistical approach given in [Sirota 1965].

While it is intended that existing Human Computer Interface techniques will be used in this application, it was hoped that some entirely new techniques might be explored. The concept of “Elaboration”, as in providing extra information to aid

¹¹ See Figure 39 Adding a word to the Windows Speech Dictionary.

¹² See Prevent a Word from Being Recognized

in recognition of tricky names, is explored as a possible improvement to the task of assigning names.

Relevant Research Sources

To create the voice controlled recording system, the state of the art in voice user interface design was surveyed for ideas and techniques that make hands-free speech recognition based user interfaces effective. A list of sources and the useful ideas gleaned from each is found in Appendix H: Bibliography.

This research draws ideas from the following areas: Hands-free Command and Control [Feng], [Lepinski], [Saruwatari], Voice User Interface [Yavelow], [Schmandt], [Hartman], [Ayres]; Human-Computer Interaction in Conversational Agent Systems [Ross], [Gruenstein], Computer Supported Collaborative Work [Colblath 2000], Multimodal Interfaces [Dumas], Natural Language Understanding [Baker] and Dialog Systems [Hindus] [Bohus, 2005,6,9].

Of course the research relies heavily on foundations in Automatic Speech Recognition [Lee], but since the core signal processing is being done within the speech recognition engine, there is not a lot to differentiate them.

We shall see, however, that aspects of the underlying recognition engine can be utilized when we investigate the idea of Elaboration. General references for Speech Recognition include [Baker et al. 2008-9] and [Jurafsky & Martin]. From this wide range of research topics, each area could provide ideas to be used to enable and improve music recording workflow. The challenge would be in effectively evaluating which ideas are the best ones to pursue. Chapter 2 examines some of these ideas and explains why some ideas are pursued in this research, and why others are not.

The areas of importance for combining voice commands with traditional control devices include:

Voice Navigator for music creation [Yavelow]

“Now with the Voice Navigator, you can just sit at your mixing console and issue all the track muting/soloing, rewind , sync, start playback at ... , etc. commands with your voice, never taking your eyes off the mixer's meters nor your hands off the faders.”
[Yavelow]

Cannon ShutterVoice software for voice control of camera, by Scott Forman¹³ is like starting and stopping a recording remotely by voice command. The feature

¹³ [http://www.robgalbraith.com/bins/content_page.asp?cid=7-9318-9769]

not provided by ShutterVoice is the ability to name photos taken and viewing them by voice commands. This would allow photos to be tagged and the decision of whether or not to take more pictures based on the success or failure of those already taken.

Text Editing by Voice [Klarlund], Programming by Voice [Hubbell], Aircraft Control [Reed 2004], Voice control of machinery, for hands free operation [Sepe 1999], Integration with a spreadsheet application [Gorniak], Rough and Ready meeting notes and summarization [Colblath 2000] each provided insights into voice control in their respective domains.

This demonstration system is tested for effectiveness in the support of recording workflow, and then is improved in the way that will be most useful and the performance is again measured to gauge the effectiveness of the chosen enhancement. Then suggestions are made for how it might be extended into other areas, such as video recording, or recording lectures and presentations.

Organizing Principles

To begin, the following are some organizing principles for the concept of voice command controlled music recording.

First Principle: Only one person provides voice commands to the system, and thus, only one microphone and corresponding signal path into the system's voice recognition module. See Figure 1 for a diagram of the signal path for voice commands and for music recording. For music recording sessions we can refer to this person as the "producer". For a video shoot, the person calling the shots is called the "director". A USB headphone microphone is used for capturing voice commands, thus providing the nearest proximity of microphone transducer to the producer's (or director's) voice. This allows voice commands to be picked up and digitized as near to the speaker's vocalization as possible, for maximum signal to noise ratio going into the speech analyzer system.

A second organizing principle is that audio tracks are named according to their content. For example, if a track contains a piano sound, then it should be named "Piano" or a similar descriptive name. If it is one of many piano tracks, then each can be appended with a number for easy identification.

A third organizing principle is the concept of "Modes of Operation". The system is expected to respond to voice commands within the context of the workflow of recording. There are distinct events in the process of making recordings, namely the start of a "take", the "action" of the scene or performance, the ending of a take, and finally rendering judgment about the take and whether or not to repeat.

The goals of the research in this paper are to explore, by integrating existing technology, the ability of a computer system to keep up with the creative intentions of a musician, without causing the flow of the session to veer into technical difficulties.

Survey of Prior Work on Voice Command Systems

Speech recognition research and development has yielded four categories of speech recognition systems, given in the columns below, each containing source references of that category:

Dictation	Dialog Systems	Command/Control	Conversational
Speech to Text	Telephone Menus	Robotics & Hands-free	Artificial Intel.
Sphinx [Lee]	Bohus	Chun-Liang	Lemon, Gruenstein
Kai Fu Lee	Gruenstein WAMI City Browser,	Feng	
Wessel	Hindus	Lepinski	Baker, et. al. 2009
	Lopez-Cozar	Saruwatari	Ross, et. al. 2004
	McTear	Schuricht	
	Ross	Singh, Chauhan,	
	Schnelle	Yavelow	

Table 1 Speech Recognition Research Sources (Authors) by Category

This research sought out ideas from each of these areas, but falls primarily within the Command & Control category.

In the article “Voice Navigation for the Macintosh Musician” [Yavelow], a system for controlling music production using voice commands was described in terms of what could possibly be implemented with speech recognition technology at that time. An investigation into subsequent uses of that particular product revealed a shift in focus toward hands-free Radiology annotation. It is unknown exactly why, or to what extent, voice commands have been used for music recording over the intervening years. AppleScript VoicePak using iListen speech recognition to control GarageBand was a released product, but without a published review since the fanfare surrounding its release.¹⁴

However, the process of recording music has enjoyed great strides using advancements in digital technology, particularly in the ease and precision of manipulation and duplication. An entire cottage industry of music producers has been enabled by the Digital Audio Workstation and audio editing software products from companies such as Avid, Digidesign, and Apple. Thousands of project studios employ these systems every day. There has been some

¹⁴ Software Review by Ellyn Ritterskamp: <http://www.atpm.com/12.09/ilisten.shtml>

groundswell of demand for voice commands to control these systems by visually impaired persons, yet nothing has yet become commercially available specifically for voice control of music recording.

The use of computers for managing experience, including collaboration, note taking, summarization, and later retrieval, is very important to information management applications [Hindus 2000]. As a collaboration effort, music recording provides an example of a process rich in interaction, structure, experimentation, revision, and improvisation. If this process can be streamlined using voice commands, then musicians and producers will benefit, but the ideas learned can be transferred to other collaboration processes, such as lecture/lab, meetings and more generally in human/computer interaction.

Other aspects of system design beyond signal processing algorithms can extend the effectiveness of Speech Recognition, as well. In perhaps one of the more historically significant papers on Voice Command Interfaces [Schmandt & Hulteen 1982] describes a multimodal Intelligent Voice-Interactive Interface system called “Put That There”, in their often cited paper entitled “*The intelligent voice-interactive interface*”. “Put That There” accepts voice commands in conjunction with gestural actions such as pointing to items or locations on a

display. The associated gestures informed the system in a more natural and effective way than by voice or gesture alone.

[Rudnicky, 1989] in “The Design of Voice-Driven Interfaces”, describes the issue of Language Design: deciding the grammar, range of spoken commands and what to do for each, comprising the Core Language, which does not change with use. Protocol Collection experiments are intended to reveal the natural spoken commands that users will prefer to say in accomplishing each task the system facilitates.

While language design considerations may make a system more effective, it may not be considered a flexible system by users speaking naturally. A system in which the user can specify the preferred voice command to be used for a given action is indeed a more flexible system. [Gorniak & Roy, 2003] describe such a system in “Augmenting User Interfaces with Adaptive Speech Commands”. The system is initially controlled only by mouse and menu commands, just as all graphical user interface software. But it has the additional feature that if the user utters a voice command along with performing an action, the system thereafter responds to that voice command by performing the action performed when the

command was originally given. In this way, a user can train the system to respond to voice commands.

In “Understanding without Formality: Augmenting Speech Recognition to Understand Informal Verbal Commands”, [McCauley], the user need not say the precise command exactly the same way every time. This is allowed by resolving variations in command names as they are clarified. Latent Semantic Analysis is performed on the continuous speech-to-text translation, to determine whenever the user intends a new variation of a command to be interpreted as an already existing command. If a speech command is semantically equivalent to one already defined, but does not already appear in the command list, then the equivalent command is issued, and the command set is expanded to include the new variation of the command.

In “Flexible Shortcuts” [Nakano, 2008], the same philosophy behind keyboard shortcuts for menu commands is adopted in a dual approach to voice commands, to provide 1) commands for ambiguity resolution (when the context may constrain the use of certain commands), and 2) commands for exploration (when the user does not know the commands available in the current context).

The proposed interface attempts to resolve the “what can I say?” problem present in other Voice User Interfaces, by allowing the user to discover what is possible in the given context

These techniques address the issues surrounding the use of menu commands as voice commands, meaning the direct translation of a menu command into a voice command. In this thesis, I also wanted to go beyond the direct translation of menu commands into voice commands. I do this by focusing on mouse-click commands instead, and to do mouse-click commands by voice, the user must refer to “clickable” items by name, such as “Mute the Piano Track”.

Speech recognition technologies now appear to be mature enough and machine learning techniques numerous enough, that this endeavor seems quite possible. What we needed to measure, however, was just how effective voice command control of music recording will be initially, and how much it can be improved by various techniques.

Brief Discussion of Speech Recognition Technology:

Microsoft released Windows Vista complete with Speech Recognition as a standard feature. It allows voice commands to be used for hands-free computing in areas such as text processing, internet browsing, media playback and photo viewing. The .NET framework version 3.5 contains a set of functions which support the creation of a grammar (a sequence of words) and semantics (their associated meaning). The following table shows many of the key .NET functions and their use in this research project:

Call	Purpose
Choices.Add()	Add an element to a set of possible choices
GrammarBuilder.Add()	Add a set of choices or node in tree to grammar
GrammarBuilder.AppendDictation()	Add a slot to recognize any word
RecognizedPhrase.SemanticValue	Key to recognition result
SpeechRecognitionEngine.loadGrammar	Load a grammar to use
SpeechRecognitionEngine.Recognize	Start sending recognition events

Table 2 .Net System.Speech.Recognition highlights

Speech Recognition comes in several forms, each serving a specific purpose:

Dictation Speech Recognition is for converting speech into text. It must support a large vocabulary, and generally it must also support multiple speakers, although some training for each speaker is generally accepted.

There is speech recognition for dialog systems, such as telephone support to book travel or movie tickets, order things, etc. This type of speech recognition provides prompts and expects responses that pertain to prompts. It must support multiple speakers with a wide range of accents, without having any training from the specific user's speaking patterns. The vocabulary should be acceptably large, given the choices the system must offer, and the dialog proceeds in a fairly well structured workflow.

And then there are more generalized command and control systems, for example to control robots or other electronic devices. The vocabulary requirements are not as great as with Dictation; and training for specific users is acceptable, also as in Dictation. The flow of acceptable commands is predefined in grammar structures, and speech is confined within established contexts, as in most Dialog voice recognition systems. While ideas can be drawn from Dictation and Dialog systems, the work of this thesis falls under the Command and Control type of speech recognition.

Since the decision to use the freely available speech recognition engine built into Windows 7/Vista, there is no real need to dig into the details of how it works, except to note that it relies upon a statistical language model and analysis of N-

Grams for improving the recognition accuracy. This means that the individual words initially recognized are analyzed in relation to each other, and are adjusted according to the likelihood that they appear together.

The language model involves figuring out what words are likely to follow other words, and using that as a way to improve recognition accuracy. "The word 'empire' will be followed by the words 'state' or 'strike' [as in The Empire Strikes Back] more often than it is followed by the words 'diverse' or 'guava,' [Yegulalp, quoting Amir Mane]

The N-Gram analysis of words is utilized in the “Elaboration” technique of allowing a user to add information about a name being assigned to an entity, as a means of increasing the likelihood of correct recognition of the intended name.

Research Questions

Can Voice Commands be effective in controlling Music Recording?

Can the flow of recording basic tracks and overdubs be accurately controlled using voice commands rather than keyboard and mouse? What prevents the direct translation of voice commands into menu commands from being sufficient for widespread adoption? Why have earlier efforts not succeeded? Can recording software operation be effectively streamlined without stammering retries and frustration, resulting in an annoying reach for the mouse in order to make the computer respond correctly? Can event handling be nimble enough to keep up with changing demands of an impatient and egotistical producer? And,

is recognition accuracy good enough, or can it improve enough over time, so that the system is deemed valuable and not an unfortunate waste of time?

Can a simple mapping of existing commands available in Audacity into corresponding voice commands be sufficient to allow efficient recording? The answer to this question was determined to be negative. It was necessary to go beyond a direct mapping of mouse and menu commands and to allow items to be named and to be referred to by name. The feature of naming tracks and then issuing track commands on named tracks by name, was implemented and tested at length, and improvements were made and also tested. The naming and referring by name paradigm could be extended to refer to regions of time (such as song sections) or to resources like special effects plug-ins (like Reverb, Pitch shift, etc.).

Can the Chosen Application Framework be Sufficient to Obtain Results?

The choice of Audacity for audio recording and Microsoft Speech Recognition for voice commands has been made based on the following decision process:

Audacity is open source: This means that software changes can be (and were) made to surmount obstacles to progress, if necessary.

Audacity supports remote scripts to enact many commands, both mouse clicks and menu commands. Remote scripting allows the voice command event handler to be a separate program from the audio recording program, which may

speed development time. However, there is a drawback in that the status of the recording system cannot always be accurately known within the voice command handler (in order to know the current context), and it is uncertain whether accurate status can be communicated back to the voice command program from Audacity. Since the user can make changes directly in Audacity, the external program cannot always be guaranteed to be synchronized with it.

The SayPlay program, developed as part of this research, is the voice command handling module. Within the SayPlay program, voice commands are handled, formatted and dispatched to the audio recorder, while logging event information for accurate measurement of system performance. SayPlay can get accurate status information from the Audacity program, but since the user can make changes directly in Audacity, the SayPlay program is not guaranteed to always have the most up to date status.

Can the naming of entities and subsequent references to them by name be generalized into a technology that can be used elsewhere?

Can macro commands be created by users, consisting of a sequence of commands? Can an entire process or sequence of commands be given a name and enacted whenever that new named command is uttered?

Areas where this might be useful are: Robotics, where objects can be named to perform actions upon them.

Answering these questions:

This thesis seeks to answer these questions by creating a prototype software application supporting audio recording and playback by voice commands, and by measuring the effectiveness in actual music recording scenarios.

Then, techniques are devised to improve the command performance accuracy. Improvements are measured in terms of the measured increase in command performance accuracy.

Details of the process of naming tracks and referring to them by name are explored, and new techniques for more effectively naming tracks and referring to them by name are investigated.

CHAPTER 2: CONTROLLING THE RECORDING PROCESS BY VOICE

Chapter 2 is a discussion of techniques and technologies that are used in this research to understand the problem domain and measure effectiveness of proposed solutions.

Traditionally, recording is done by a team of studio professionals directed by a record producer. The role of the producer is much like that of a movie director, who calls out “Roll Sound, Lights, Camera, Action” to his team of technical personnel. The team of studio professionals required to record music generally consists of a lead recording engineer who controls the mixing console, an assistant, and/or a tape operator, to control start and stop the tape recorder, and the Producer and musicians.

With the advent of digital audio production systems, the mixing and recording features are combined into a single hardware and software entity called a Digital Audio Workstation (DAW), for recording, editing and mixing multiple tracks of audio.

Laptop computers are now fast and powerful enough to record a music track along with many tracks all playing at once, allowing many of the functions in a console and multi-track tape recorder to run on a table top.

But, using DAW software is not a hands-free process, allowing musicians to perform on their instruments without having to reach for a mouse to start and stop recording. Having to use one's hands to control record starts and stops by mouse click is detrimental to the workflow both practically and creatively.

Defining the Problem

With a set of voice commands to control recording transport functions such as Play, Stop, Record, Pause, Skip Forward, and Skip Back, a musician or producer could overcome the problems mentioned above, and become more productive and satisfied. Audacity is an open source program which provides many features of a Digital Audio Workstation, and is shown below in Figure 4 Audacity waveform display of audio tracks.

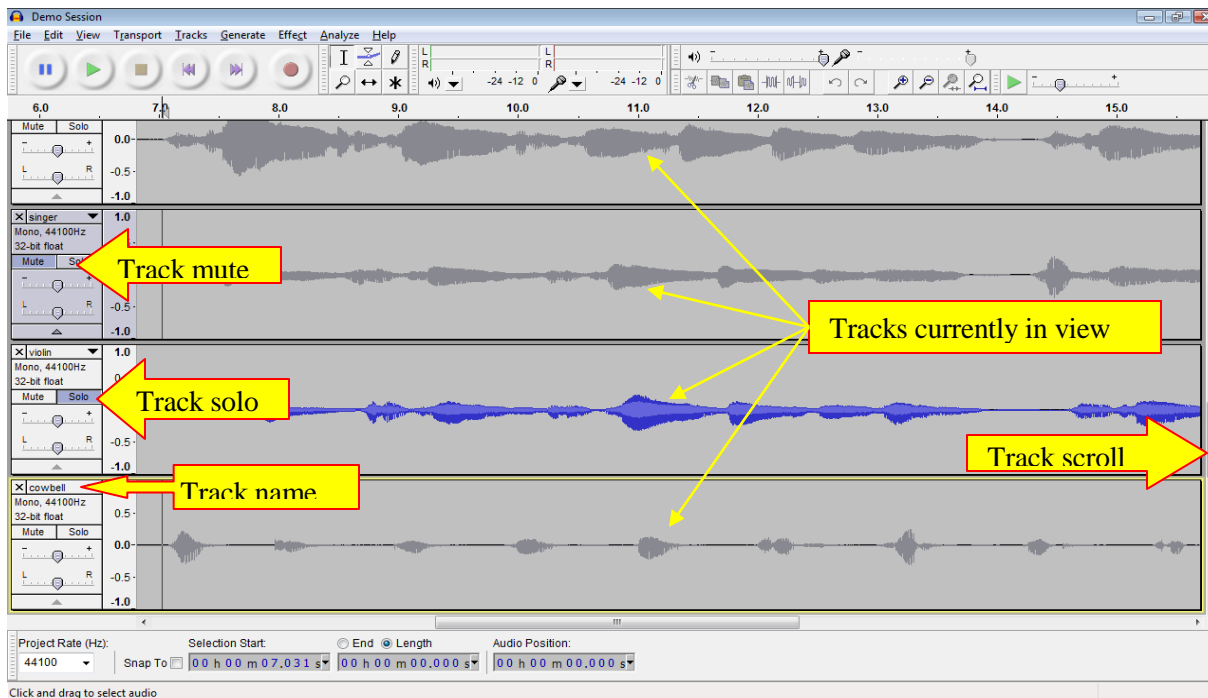


Figure 4 Audacity waveform display of audio tracks

I developed a program called “SayPlay” to provide basic voice control over the functions in Audacity, and carried out performance measurements on the effectiveness of using SayPlay to control recording in Audacity hands-free.¹⁵

¹⁵ See Appendix B: The Demonstration System, for a detailed description of SayPlay and changes made to Audacity to support voice commands.

Experimental Test Setup

Test Setup: A USB headphone is used for all experiments, so the microphone is close to the person speaking, rather than mounted on a mic stand, or built into the laptop. It is very important to use the headset microphone for the best acoustical signal to noise ratio.

The diagram below shows the components of the system: a headset microphone for input of voice commands, which was connected to the laptop computer by USB. It could also have been wireless. The other component next to the laptop computer is a USB audio interface, for connecting instruments or microphones to record musical instruments or voice.



Figure 5 Recording equipment used in experiments

The Windows Speech Recognition facility requires that the microphone be initialized for the user before using speech commands. This process is started whenever a new speech user profile is created. Therefore, each test subject will perform the microphone setup right after their user profile is created. Also, each test subject will perform at least one Windows Speech Recognition training session.¹⁶

Windows Speech Recognition (WSR) Issues

Windows Speech Recognition (WSR) provides two primary modes of operation: Command mode, and Dictation mode. One of the challenges of effective user interface is distinguishing between the two modes. For example, if a user is in dictation mode, say, dictating a report, and then they wish to switch to another application. This requires a context switch into command mode, to recognize the command to switch applications. My research doesn't require delving into this distinction, because I use only the command mode. However, dictation speech recognition is possible to use from within a predefined grammar, even for commands.

¹⁶ See Appendix C for the procedure for setting up and training Windows Speech Recognition.

Keywords and Choices within a Grammar

WSR provides the means to define command keywords each with an associated semantic result, which is returned with recognition events. Beyond this, it is possible to define complex grammatical structures, with numerous selection points, each defined by a set of choices. As an example, consider a telephone based pizza ordering service.

Wildcards within Choices in a Grammar

In addition to the pre-defined keyword choices and grammar structures which can be defined, it is also possible to define choices which are not pre-defined, and are included in the desired slot of the grammar as a “wildcard”. This is where Dictation speech recognition comes into play.

Dictation Speech Recognition within a Grammar

Dictation speech recognition can recognize words and phrases which are outside of the pre-defined grammar structure, so that, for example, the grammar that recognizes “I’d like to travel from Denver to Timbuktoo” can fill the destination slot with any name imaginable (in this case “Timbuktoo”). We will see that recognizing names using dictation speech recognition to fill the slot of the desired name, as in “Name this track *digeridoo*”, can work, but that adding ‘digeridoo’ to the grammar as a choice works even better.

The Windows Speech Recognition (WSR) Speech Dictionary

WSR provides features for adding words to the Speech Dictionary, as described in the section: Appendix D: The Windows Speech Dictionary. If several attempts fail to assign a name, and it is a rather obscure name or pronunciation, then it may be necessary to spell out the name, or to add the word to the Speech Dictionary. If a rather common word is continually mistaken for another word, it is possible to prevent the recognition of that other word, using the Prevent Dictation function described in: Prevent a Word from Being Recognized. These features expose an underlying set of data structures, the Speech Dictionary itself, and a list of words to ignore. These are depended upon by each grammar that is currently loaded into the speech recognition engine.

Grammars in Speech Recognition

SayPlay uses a different grammar structure for each command family required. See for the class diagram of SayPlay command families. For example, the keyword commands which translate command phrases like “Play”, “Stop”, and “Record” directly into commands sent to Audacity. The TrackCommand grammar requires more detailed structure (see Figure 15 Grammar structure for issuing track commands by name), to handle many commands being applied to many named tracks, with some versatility in phrasing.

User Interface Considerations:

The primary focus of this research is to provide a user interface which is responsive and completely hands-free. Therefore it must be accurate and flexible

enough that creative flow of the process is not hindered. Considerations in this regard include

Timely feedback:

Timely feedback to the user regarding the state of the system: The context will help govern the set of possible commands, for example if we're recording basic tracks, then "name this one 'Alternate' " refers to the last recorded set of tracks, whereas if we're recording overdubs, then "call that take, 'Alternate' ", refers to the last recorded track.

Ambiguity resolution

When the system cannot decide what to do with an utterance, there are two things it could be: confusion over what the user wants to do, or that the user isn't intending that anything be done because the detected utterance was not a command at all. In the first case, the system may be required to ask the user for clarification. In the standard usage of Microsoft Speech Recognition, a dialog box appears showing a numbered list of words, and a choice is made by number.

"Are you talking to me?"

Handling spurious words or comments directed away from the system: Whenever an utterance is not meant for the system, the system should simply not respond,

but it should simply note that the utterance is interpreted as spurious or not directed at it.

Addressing the computer before any commands, using a keyword, such as “Computer”, or ideally a name assigned by the user. This method of gaining the attention of the system was employed in the popular science fiction television series Star Trek, and would be familiar with most users.

Microphone Mute Switch:

The USB headphone used in the experiments features a Mute switch, to turn the microphone on and off. This feature is very handy for preventing the speech recognition system from mistakenly reacting to speech which is not directed to the Speech Recognition system.

Ready for Action:

Interruptions to the recording process, such as a broken string on an instrument, a necessary break for the performers, and so forth, should not send the system into an unknown state, but would retain a system ready state for recording the next take, simply by muting the headphone microphone, and preventing the speech recognition from getting any speech directed elsewhere.

Audio Loudness:

Loudness of user utterances can sometimes create a reverse correlation to system performance. For example if a user is not understood on the first time or second attempts, they may have a tendency to speak more clearly, but also louder and louder each time. Speaking loudly only causes distortion in the signal, making it more difficult to process and recognize. A preprocess which detects this distortion along with increased signal levels, might be used to detect user frustration, and to provide some gentle calming feedback in order to elicit a clear but softly spoken (and undistorted) voice.

Track Management:

Track management is important for building a rich multi-instrumental recording from individual takes recorded against successive playbacks of the basic tracks. It was soon discovered that only having basic commands for transport (Play, Stop, Record) were insufficient for allowing hands free workflow. Because the user needs to silence and activate individual tracks, using the Solo and Mute controls, there needs to be a way to refer to them. Initial attempts found good results in muting and soloing tracks identified by a given track number. Track 1 is the top track, the first one recorded, followed by Track 2 beneath it, and so on for as many tracks as there are in the session. While recognition accuracy was

reliable in allowing the user to issue track commands by voice, it was quickly decided to be very impractical because it required the user to remember the audio contents corresponding to each track number. And, as shown in Figure 4 Audacity waveform display of audio tracks, there is no visible track number indication for each track, but there is a track name. And the track name can be quite descriptive of the audio contents of the track, for example: “Piano”, or “Lead Guitar”.

Referring to tracks by name is ideal, since the name is semantically related to the contents of the track. For example, a track may contain a voice recording, and the name can be the name of the person whose voice is contained on that track. Alternately, a good track name is the name of the musical instrument recorded in that track.

Named Tracks:

The operation of assigning names to tracks, by which they are subsequently referred, has become an important addition to basic commands, because the user can now issue commands on tracks, whether or not they are currently visible on screen. And it is much easier for a user to recall a track by name than by number. The Activity Diagram below shows the process of assigning a name (banjo) to

the track that has just been recorded. Note that the most recently recorded track is the last track (shown on the bottom) in the session, and thus, we get that track number when assigning the name.

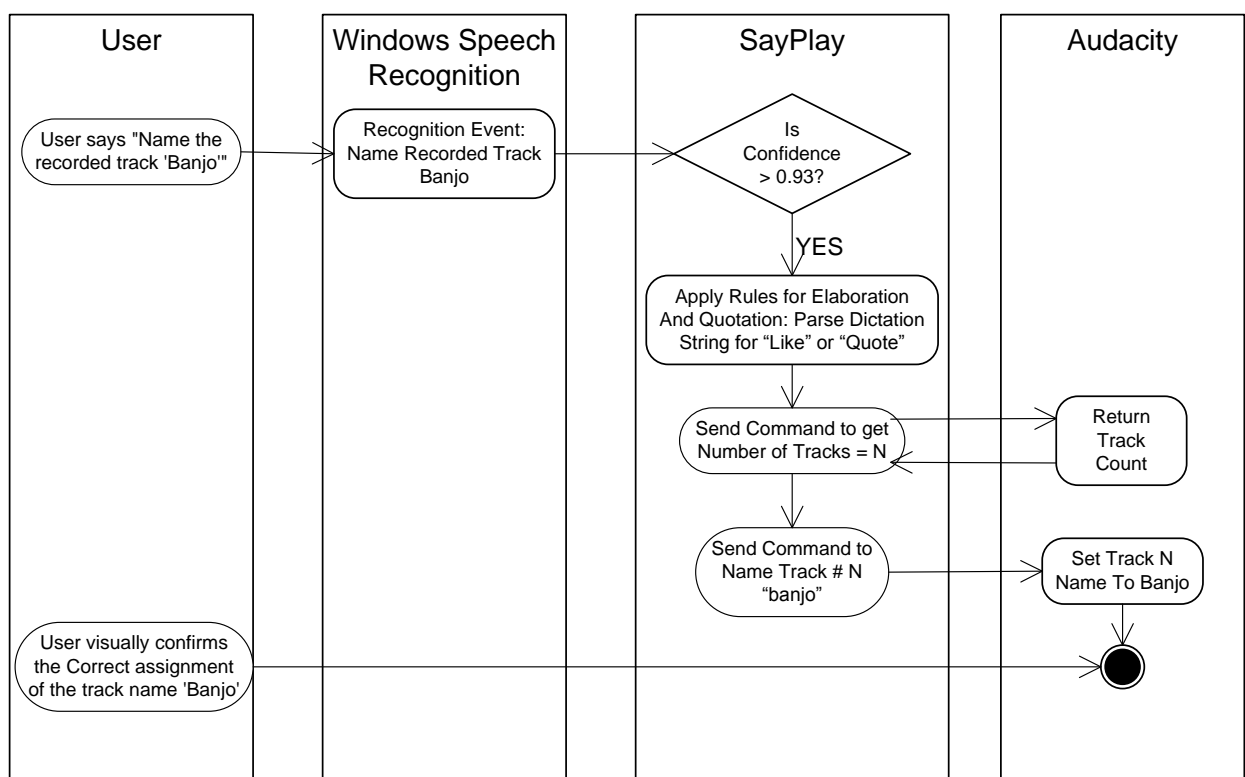


Figure 6 Activity diagram for assigning “Banjo” to the recorded track

Once the user has assigned the desired names, and confirmed their assignments visually, or by successfully issuing commands upon them, he may load the names into the Track

Command Grammar, as choices for the object of the commands.

The process of loading session track names into the grammar is shown in the activity diagram below. This process alleviates the need to continue to recognize track names using dictation speech recognition, since the names are known and can be loaded into a grammar as a list of choices for where to apply a command.

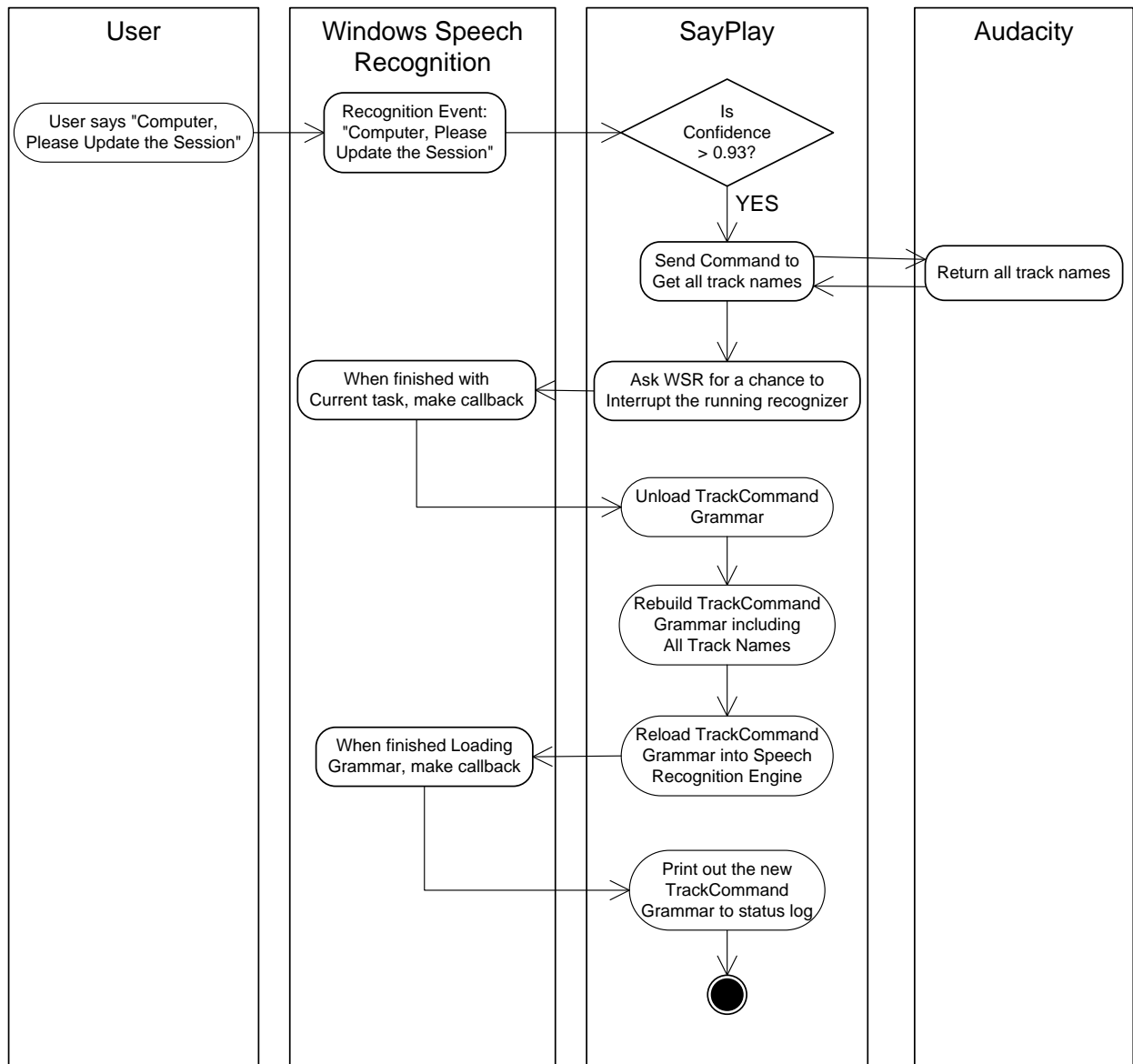


Figure 7 Activity Diagram for loading track names into the loaded Grammar

For activity diagrams to describe other operations, please see: Figure 29 Activity Diagram for naming a track by spelling out the name, Figure 30 Activity Diagram for naming a track using “Elaboration”, Figure 31 Activity Diagram for the Track Command to Play a Named Track, and Figure 32 Speech Recognition microphone setup.

Factors Affecting Recognition Accuracy:

Audio Signal Problems

The SayPlay interface provides an indicator that displays the speech recognition input signal status (`Event.AudioSignalProblem`). The following are the possible states the Audio Problem indicator can take:

Too Noisy: There is too much background noise for engine to process.

No Signal: No input signal detected.

Too Loud: The input signal is too loud for the speech recognition to process.

Saturation distortion causes lower recognition accuracy.

Too Soft: The input signal is not loud enough for the speech recognition to process. Initially, whenever SayPlay was launched, the microphone input level would automatically be set to a setting of 2 (on a scale of 1 to 100). It is strongly suspected that the initial setup of the microphone was too loud, and the low

setting was set to compensate. However, it was discovered that as a result, basic speech commands such as Start and Stop, were recognized with rather poor performance. Correcting the microphone setup resulted in a higher automatic microphone input level setting, and in better speech recognition performance.¹⁷

Too Fast: The input signal is too fast for the speech recognition to process.

Too Slow: The input signal is too slow for the speech recognition to process.

Microphone placement:

Mic should be no more than an inch or two from the mouth of the speaker.

Having it too far away, such as the lap-top computer's built-in microphone will result in a greater influence of room noise. If it is too close, it may result in too much sound from disfluencies and non-verbal gestures, such as clearing the throat, licking the lips, or the "breath after" problem.

Phrasing and the "Breath after" problem:

Sometimes articles such as "The", "a", "and", erroneously appear before or after the intended name of the track. Usual examples occur when taking a breath

¹⁷ Figure 9 Baseline performance test, mean WSR Confidence

following a phrase, such as in “Name this track guitar *-huff*” (take a breath), being recognized as “Name this track guitar and”.

Accent is an important factor in the initial success rate using Speech Recognition for command and control. Even after completing two training sessions, the test subject with the strongest accent had very poor rates of success. It was only after repeated attempts to use voice commands did they even begin to succeed, and the meager initial success improved more slowly than for native American English speaking test subjects¹⁸. The impact of accent was also discussed in [Schuricht].

Potential Problems

Problems are in order of severity, with the worst possible problems listed first.

Destroying data: If a misrecognized command results in destroying data, such as a completed take, or worse, an entire session, then the software will be deemed a problem.

¹⁸ Figure 11 Cumulative successes for 5 test subjects

Missing a take: If the software results in the failure to record a performance, then this is also a problem. For example, singing a rendition of the classic Motown hit: “Stop, in the name of love”, the “Stop” would cause recording to halt, failing to record the take.

User frustration: If the user must repeat a command more than is comfortable, then alternate means of surmounting the obstacle which speech recognition has become are sought. The author believes that it helps to be able to rephrase the command when repeating it, as humans do when a statement is not heard well enough to be understood.

Ambient noise, music or other sound: This can adversely affect the ability of the Speech Recognition system to function correctly. This issue is minimized in the experimental setup by using microphone positioned close to the speaker’s lips. Experiments were not carried out on the interaction of ambient noise, music listening levels or the like. This problem is considered to be tangential to the more important issue of supporting the recording workflow by naming tracks.

Reaction time (human and machine): If reaction time is too slow, then assertions from the user will not be matched with reactions by the system, and the user may repeat a command while the system responds to the first, and both, or gets into an undesirable state. This can be critical when the user attempts to identify a specific moment in time. For example, during playback the user may wish to mark the beginning or end of a selection, by first pausing playback at the specific point in time. The delay in recognizing and issuing the pause command could cause inaccuracy in the setting of the desired point. The user might compensate for this delay (having gotten used to how long the delay is), but if the system improves its response time, or lags due to sporadic overload, then the compensation itself may require compensation.

Homophones: (words that sound the same, but are spelled differently):

Distinguishing between two words that sound exactly the same can only be done based on the context in which those words are spoken. If the word is used as a name, then the context will be necessary to determine which spelling is correct.

Homographs: (words that sound different but are written the same way):

Homographs can only be represented in one way as text, but have different sounds, for example “Record” “Tear” and “Read” are homographs. “Record” can be a verb (re-CORD), or a noun (REC-ord). Despite the idea that

homographs can pose a problem to Speech Recognition grammars, which are defined by text words, the homographic command “record” (used to begin recording) was found to be quite reliable, after training.

Description of Experimental Apparatus Software Design:

SayPlay

The voice control part of this system was designed and implemented in the “SayPlay” program. SayPlay can log every speech recognition event (including comments as well as commands), to a file that is created when SayPlay is launched. Speech recognition events are then analyzed statistically to show performance trends and specific failure modes, in order to find out when and why certain failures commonly occur. For more on SayPlay, see: Appendix B: The Demonstration System

Audacity

Audacity is a multi-track audio recorder and editor program with a scripting interface so that other programs can issue control commands (to Audacity), for

recording and playback of audio, as well as other actions. Audio recordings are organized in horizontally linear waveform displays, called Tracks, which have standard mixing controls, Solo, Mute, Pan and Gain.

Please see Appendix B: The Demonstration System, for the description of how Audacity fits into the software design of SayPlay.

Overview of Experimental Procedure

Experiments have been designed to exercise voice command functions and to note whether or not the system responds correctly. Accuracy is measured by statistical analysis of success rates and the attributed causes for failures.

Recording Success and Failure of Command Events in a Log File:

A single test event consists of the test subject speaking a command phrase, and then determining whether or not the command was correctly carried out.

Success or failure are verbally indicated by speaking the word “Pass” or “Correct” for success and “Fail” or “Wrong” for failure. These Pass/Fail indications are written into a log file with each associated command. The success-indicating word “Pass” was frequently mistaken for the command “Pause” by a user having a strong accent, prior to sufficient training.

The experimenter may also append a comment about an event. The comment must be stated immediately following the Pass/Fail indication, because there is only a limited time for the comment to be accepted, after which spoken words go back to being considered commands. Comments can also be added to the log file without association to particular events, simply by prefacing it with the keyword “Comment”. This is useful when indicating the purposes of an experiment, descriptions of external events that could not otherwise be detected, such as changing a system setting, or taking a break for lunch.

Voice commands appear in the left column, with the middle column showing a numeric confidence value (a decimal number between 0 and 1 that is returned by the Windows Speech Recognition engine with each recognition result).

UnMute can a track	0.9530823	Wrong name
UnMute the track named piano	0.9220377	Correct
Mute the guitar track	0.9215692	Correct
UnMute the guitar track	0.9482349	Correct
Mute the drums track	0.8997931	Wrong nothing happened
UnMute the drums track	0.9508225	Correct
Solo the piano track	0.9323536	Correct

Figure 8 SayPlay logfile excerpt

The TAB delimited columns in the log file are from left to right: Spoken command, Confidence value returned by the speech recognition engine, Pass/Fail determination followed by an optional comment.

The log file is created on startup of the SayPlay application, and is saved during normal operation whenever events are added to the file.

Voice Command Failure Modes

When commands fail to execute, the failure is attributed to one of the following:

Low Speech Recognition Confidence: The command is correctly recognized, as shown in the Log File, but the confidence value returned by Windows Speech Recognition (WSR) is below threshold, so no command is sent to Audacity.

Wrong Name: The track name is misrecognized, but the command is sent to Audacity because the WSR Confidence value is above threshold. Since the name is incorrect, it is not acted upon by Audacity because no track of that name exists.

Wrong phrasing: User error, or an otherwise incorrect wording of a command.

This can be as simple as mumbling, stammering, or could be a failure to remember the correct phrasing of a command. It could also be the failure to remember that the command is not supported in the current context.

Timeout violation: A long pause in phrasing causes the recognition engine to act on the first part of the utterance, as if there were no more words to come.

False positive: An utterance which is not a command is misinterpreted as a command, with a WSR Confidence value above threshold, resulting in an unintended command being mistakenly issued to Audacity.

True rejection: An utterance which is not a command is misrecognized as one, but the confidence value returned by Windows Speech Recognition is below threshold, and so it is correctly rejected

“Breath After”: When Windows Speech Recognition is using dictation speech recognition, it is possible to misrecognize a dysfluency following a recognized word, as a mistaken additional word. This occurs where optional additional wildcard words are possible, such as within the name for naming an audio track. A dysfluency can be a quick breath inhaled, or the sound made by lickings one’s lips or clearing one’s throat. For example, the track naming command can turn out wrongly recognized as having “and” at the end, as in: “Name this track bass guitar <and>”.

Another example occurs when adding a comment following a failure indication. “Wrong I” appears where a comment can be added to “Wrong”, but instead of stating an optional comment, the user’s inhale is misrecognized as a word. A simple solution to the “Breath After” problem has been to prevent certain commonly mistaken words from ever appearing at the end of a name: “I”, “and”, “ah”, for example. However, as with any application of simple rules, there are exceptions, and those will also be prevented from occurring.

Test Plan Overview

Testing was interleaved with exploration, in order to prune the number of opportunities for research down to a manageable set. See Appendix E: Test Procedure. Test results and discussions follow, and can be accessed directly from the active links in the following table.

Test Plan						
Test Context	Basic transport commands Play/Stop/etc.	Keyword Commands (the long list)	Track commands (Solo/Mute of named tracks – many users)	Tricky Names recognition accuracy	Preventing Dictation and Loading Names into Grammar	Elaboration and Loading Grammar
Test Description	Basic transport commands Play/Stop/Pause/Record	Test each Audacity scripting interface command	5 test subjects on Keyword and Named Track Commands	Before and After tests of Theremin, Wow, Crotales and Gambales	Prevent dictation of confused names, then load names into grammar	Use Homophones as Names
Test Procedure	Table 8 Test 1 Procedure	Table 9 Test 2 All Keyword Commands	Table 10 Test 3 Test Procedure: Field Trials	Table 11 Test 4 Test Procedure: Tricky Names	Table 12 Test 5: Effectiveness Adding Theremin to Dictionary and Grammar	Table 14 Test 7: Homophone Tests of Elaboration and Loading into Grammar
Test Result	Figure 9 Baseline performance test, mean WSR Confidence	Figure 10 Mean Confidence value returned from WSR Engine	Figure 11 Cumulative successes for 5 test subjects Figure 12 Errors by category of error Figure 13 Success rates of Keywords Commands vs. Named Track Commands	Figure 16 Cumulative Success Before/After Adding Name to Speech Dictionary	Figure 17 Recognition Rates after Successive Improvements	Figure 21 Recognition Accuracy after loading homophones into loaded grammar
Interpretation	Basic Commands:	Keyword Commands	Improvement in Command Accuracy	Improving Accuracy in Referring to Assigned Name	Adding Names into the Windows Speech Recognition	Elaboration and spelling which of the Homophone pair to use as name

Table 3 Test Plan

CHAPTER 3: COMMAND RECOGNITION ACCURACY

Initial tests focused on basic commands (Play, stop, record, etc.) in order to establish a baseline of performance, and then continued to test all keyword speech commands available in the Audacity scripting interface. A field trial with multiple test subjects, using commands operating on named entities, was then performed. Problem areas were identified and improvements were implemented to address them. Further testing was performed to support the claims that the techniques implemented resulted in actual improvements.

Effectiveness of using voice commands to control music recording is largely determined by the accuracy of the speech recognition technology in use. And the factors affecting accuracy of speech recognition are many, including microphone placement, ambient noise, speaker accent, user error (not stating the command correctly), and the pace and spacing of words being spoken.

Each of these contributing factors could be analyzed in-depth in isolation. I have chosen to investigate the performance differences between keyword command recognition, and named track commands which rely on large vocabulary dictation speech recognition in order to recognize the name correctly.

Basic Commands:

After discovering that the microphone setup process had set the level to an unacceptably low value, the baseline experiment was repeated.

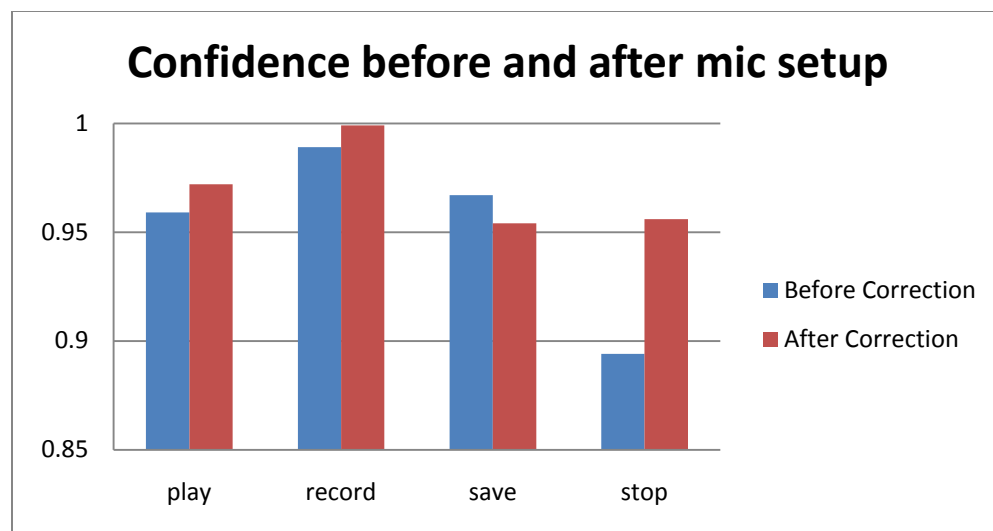


Figure 9 Baseline performance test, mean WSR Confidence

The above shows how microphone level can adversely affect recognition confidence (that value returned with each event by the Speech Recognition Engine). The average confidence for “Stop” was much lower while the microphone input level was erroneously set to a value of 2 (on a scale of 1 to 100). The test data was subsequent to setting up the microphone.

Keyword Commands:

Keyword commands are those for which Audacity provides a scripting interface, and are handled by SayPlay and sent on to Audacity whenever the Confidence value for the recognition event is above a threshold of 0.93. The confidence values for each command are averaged and tabulated in figure below. The test data comes from Table 9 Test 2 All Keyword Commands, in Appendix E: Test Procedure. Mean Confidence Value is returned from the Speech Recognition Engine for each command listed is above the threshold.

It was found that for all commands, the average confidence value returned by the Windows Speech Recognition (WSR) engine was above the threshold for deciding whether or not to act on the command. This indicates that speech recognition can be used with a reasonably high degree of accuracy when single command words and phrases are uttered during the process of recording. It does not however indicate the expected success for commands which pertain to particular tracks of the session, indicated by their names. This type of command, the “named track” command, is discussed later.

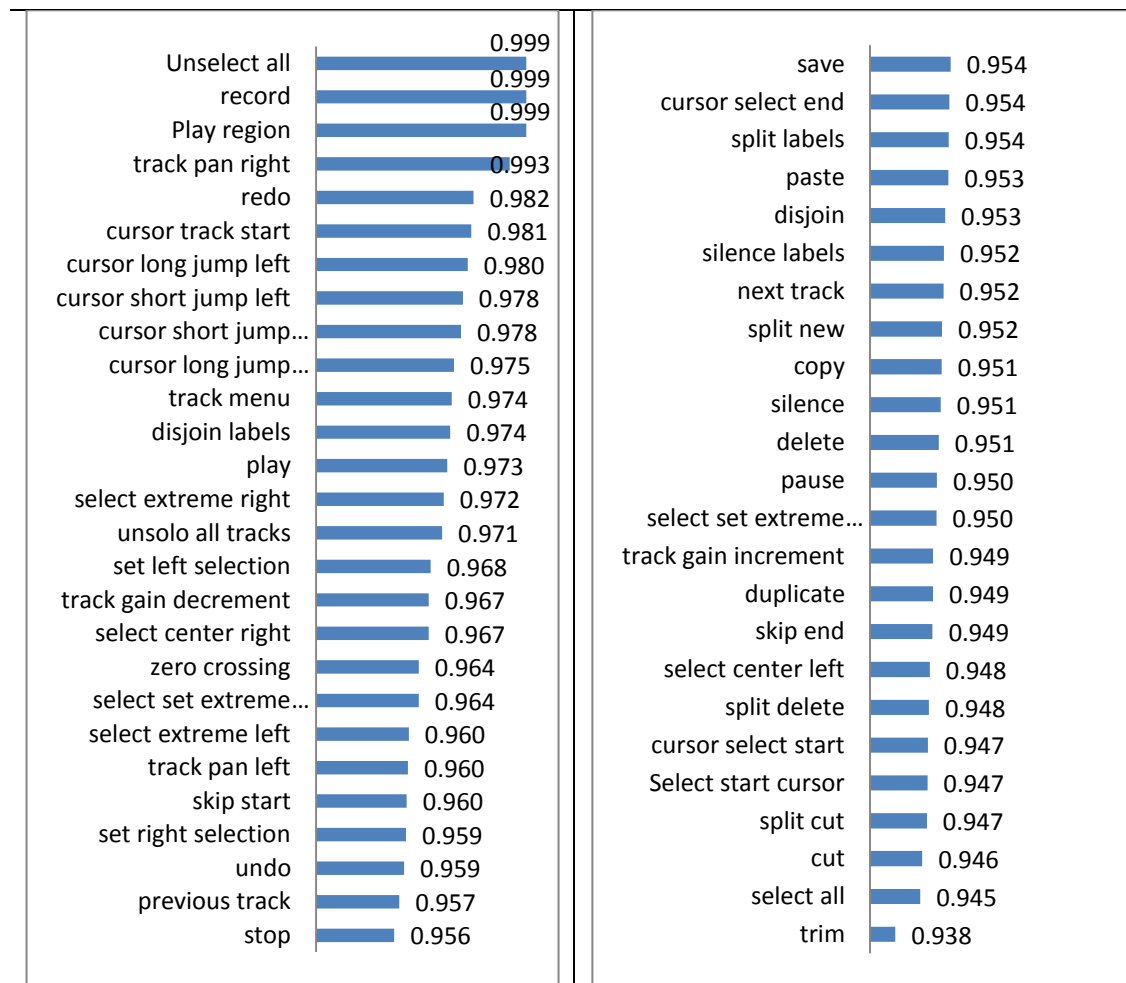


Figure 10 Mean Confidence value returned from WSR Engine

Improvement in Command Accuracy

The figure below shows the results from Table 10 Test 3 Test Procedure: Field Trials. Cumulative successes are shown for voice commands issued in a simulated recording session. These are compared against the ideal “perfect” line

if all commands were successfully executed. The differences in the rate of improvement for each test subject correlates most closely with speaker accent.

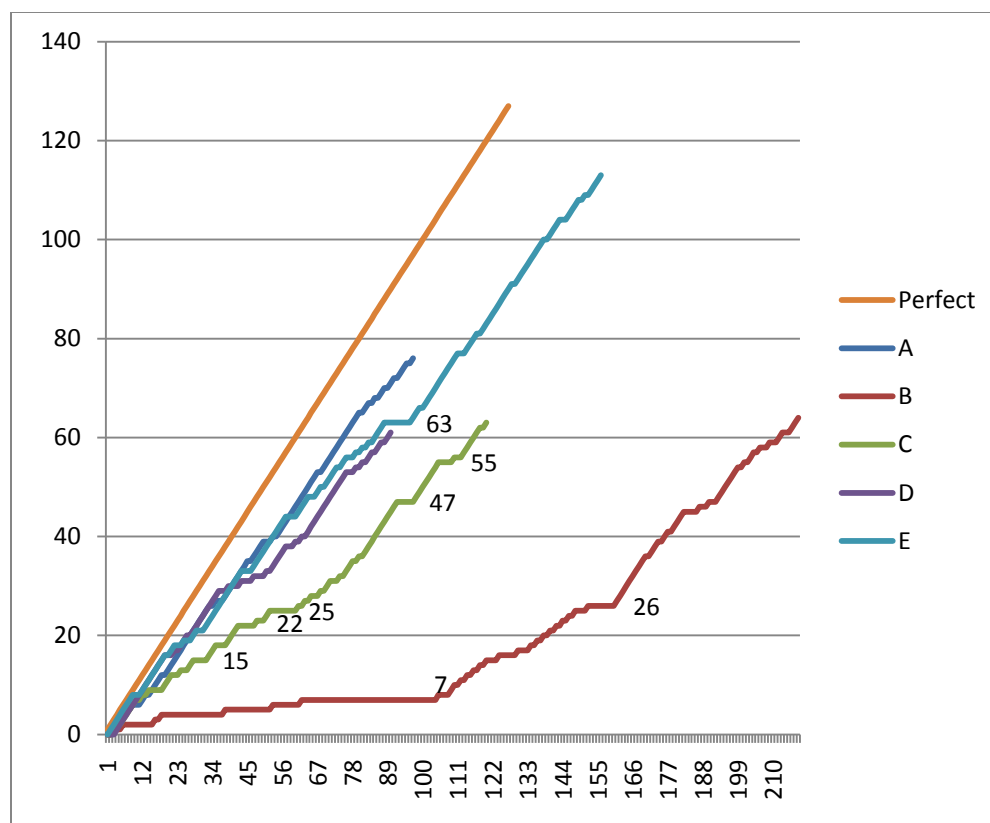


Figure 11 Cumulative successes for 5 test subjects

Point 63 is trouble with "Vocal Scat" : "Vocals get", "focal scat", "Vocal scout".

Point 25 is trouble with "Alto Sax" mistakes: "alta sachs", "up to sax", "otto sax/sachs", or "office acts". The naming command tends to favor proper nouns

as results. This is why “Alta Sachs”, or “Otto Sachs” appears. We could try altering the track naming command to “Assign this track name”...

15 and 47 are trouble with "Violins" mistaken for "Violence"

22 and 55 are trouble with "Vocal Scat" : see SayPlayLogFile_367051199.txt

7 and 26 were successful commands alternating with false positives of utterances of "Pass" that sound like "Pause"

26 is start of second session, and continued trouble with alternating false positives.

The precise source of performance improvements cannot be attributed solely to the Speech Recognition Engine, because the speaker will unconsciously change the way words are pronounced, particularly once success has been achieved. The testing was not done by professional voice talent, intentionally speaking the same way every time. If a user found success, then it's likely they will attempt to emulate the same tone, level and pronunciation when subsequently trying for successful command recognition. But, there needs to be a few successes for the user to get a feel for what works.

It is not known to what extent the demonstrated improvement in performance is due to the Voice Recognition system learning from examples, or how much is because the user has improved their articulation and pronunciation.

For the above experiments, it is helpful to attribute the causes of failures to one of the many failure modes described before. This yields the following chart of errors by category:

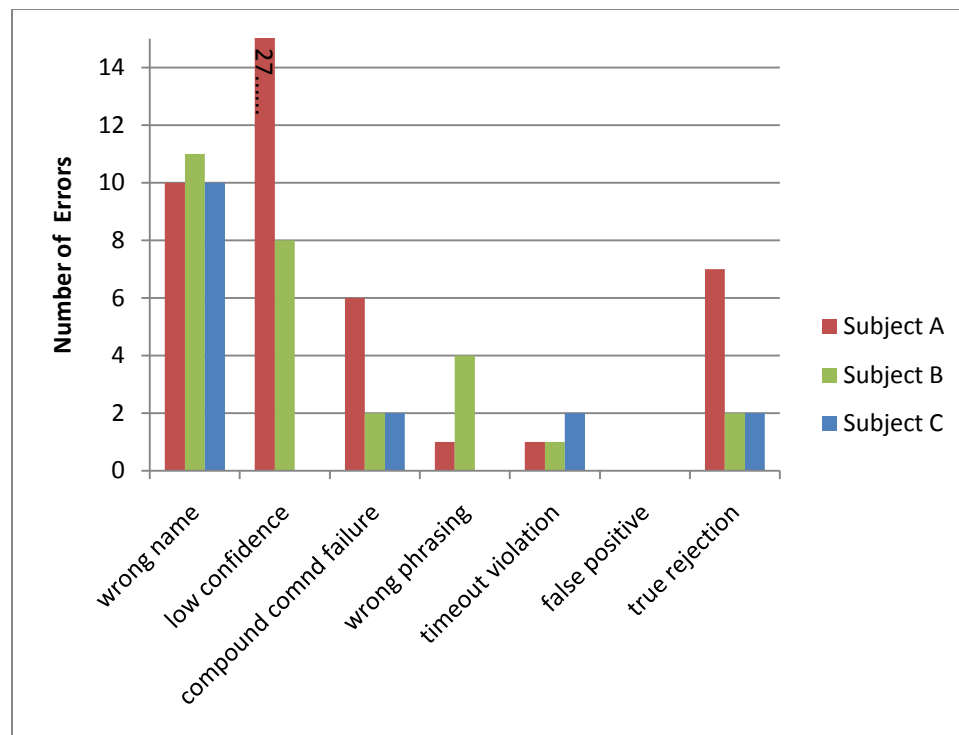


Figure 12 Errors by category of error

From the analysis of the annotated log files, the majority of errors are of “low confidence”, followed by the “wrong name” category. For a complete list of error categories, see: Voice Command Failure Modes.

To improve the low confidence performance, it is possible to simply lower the confidence value threshold to accept commands with lower confidence. This could result in more false positives, however. Or, if the user can bear repeating himself at the beginning, the confidence value does increase over time with use.

It is helpful to further examine the errors of low confidence, because within this category, there can be false positives, wrong names, and true rejections.

After several rounds of testing numerous test subjects, the event data was annotated and grouped according to whether the event was a Keyword Command (which are entirely specified by the Speech Recognition grammar) or a Named Track Command (which require correct Dictation Speech Recognition of the name). The results shown below indicate that recognition rates for Named Track Commands are consistently lower than for Keyword Commands.

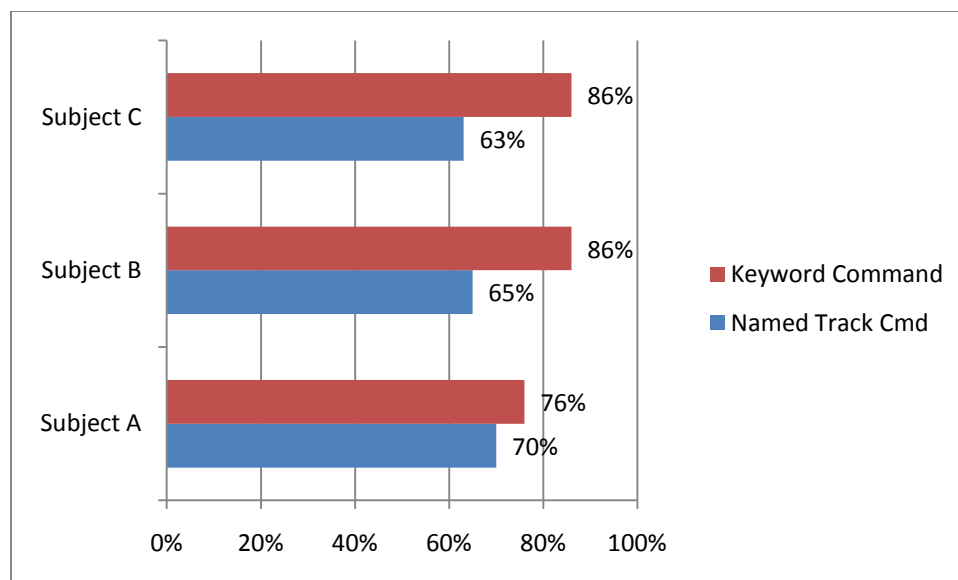


Figure 13 Success rates of Keywords Commands vs. Named Track Commands

With the performance of basic commands being higher than for named track commands, effort was directed toward improving the “wrong name” type of error. In Chapter 4 we turn our attention to improving the accuracy of Named Track commands.

CHAPTER 4: NAMING AND REFERENCING TRACKS BY NAME

One of the most likely causes of failure is getting the wrong name, as demonstrated in Chapter 3. In addition, the recognition rate was lower for named commands than for keyword commands. In Chapter 4 we examine more ways to improve the performance of naming tracks, and commands that refer to names.

Flexible Grammar Constructions

The diagrams below show the grammar structures to allow track naming (Figure 14 Grammar structure to allow tracks to be named.), and subsequent track commands to refer to tracks by those names (Figure 15 Grammar structure for issuing track commands by name).

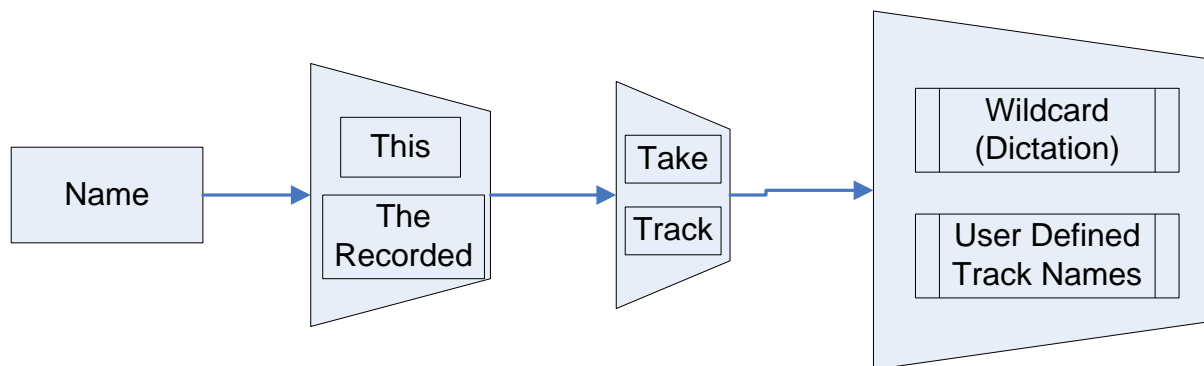


Figure 14 Grammar structure to allow tracks to be named.

Thus the recognized phrase can be simply: “Solo the guitar”, instead of “Solo the guitar track”, or even simply “Solo guitar”. Or it could be stated as elaborately as: “Solo the track named guitar”. These variations provide the user with some flexibility in how commands are phrased. Later, test results are given where the chosen phrase may affect accuracy. The user may rephrase a command if it is not carried out on the first attempt.

For example, if “Mute the Violins” fails, the user can restate the command “Mute the track named Violins”, for better results.

Note that items shown in curly braces ({...}) are optional.

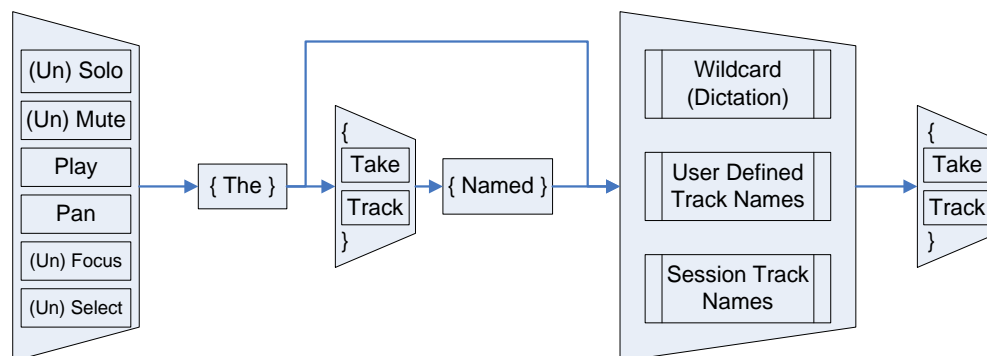


Figure 15 Grammar structure for issuing track commands by name

Flexibility comes from allowing words to be optionally present within a recognized phrase. As shown above the optional article “the” and the optional noun “track” allow a user to say any of the following: “Play the guitar track”,

“Play the guitar”, “Play guitar track” or simply “Play guitar”. This flexibility is nice because different users have been found to favor different phrases, and if restating the command is necessary due to a recognition failure, it is human nature to want to rephrase the command differently for a better chance to be understood.

This flexibility, however, is hard-wired and allows variation only as far as has been programmed. A user cannot say the following and expect it to work: “Okay, now let’s hear the guitar track without the others”, which is merely a paraphrase of the command “Play the guitar track”.

Overview of Improving Name Command Recognition

Earlier measurements showed that Name Commands had lower recognition accuracy than keyword recognition. The following techniques are used to increase the accuracy in naming tracks and referring to them by name. First, the name must be assigned correctly. To correctly assign a new name, there is:

1. Elaboration
2. Quoting the name as a phrase
3. Spelling out the name

Each of these methods for assigning names is described in detail in the sections that follow.

Once the name has been correctly assigned, there are ways to improve the recognition accuracy of named entities in commands which refer to them.

The techniques for improving name recognition, once assigned, are:

1. Load the assigned names into the corresponding grammar. These names would occupy the “Session Track Names”, when referring to the lower right box in the diagram: Figure 15 Grammar structure for issuing track commands by name. Also see Figure 7 Activity Diagram for loading track names into the loaded Grammar, for the steps for carrying out this action in detail.
2. Adding words to the Windows Speech Dictionary. See Figure 39 Adding a word to the Windows Speech Dictionary. See Figure 40 Record a pronunciation of new word added. For saving a spoken example of the new word being added.
3. Preventing confused words from being recognized. See Figure 41 Prevent a Name from Being Recognized and Figure 42 Prevent a word from being dictated.

Each of these methods for referring to names is described in detail in the sections that follow. See section 2.2 Windows Speech Recognition (WSR) Issues for a review of the introduction to these features.

Adding names to the Speech Dictionary and preventing Confused Words from being recognized are accomplished using the Windows Speech Recognition control panel. Including the assigned names as choices in a grammar loaded in the speech recognition engine is done programmatically at the user's request.

Improving Accuracy Assigning Names

Elaboration:

Elaboration of names is made by including a simple construction of "Like" or "As In" with the naming command, along with associated words that improve the chances for correct recognition. For example, if the user tries to name a track "Bass", as in the musical instrument, but instead the system recognizes the homophone "base", as in baseball, then the user can simply state: "Name this track bass like the bass guitar", or "Name the recorded track bass as in the musical instrument". The extra words utilize the statistical language model to tip the balance in favor of the correct sense and meaning of the word "Bass".

Elaboration is implemented by the positive detection of the substrings “Like” or “As In”, within the name string, and then truncating the string to retain only what comes before “Like” or “As In”.¹⁹

An implementation detail was that the key phrase “As In” became problematic in actual practice, as it was frequently misrecognized as one of the following: “As An”, “Is An”, or “Is In”. Therefore, these key phrases were also included in the rule to detect elaboration, and so they are also removed from a name when recognized as elaboration.

The drawback of this technique is that it always blindly removes any part of the string following and including the key phrases listed above: “like”, “as in”, “as an”, “is in”, “is an”. So, if the user wishes to have these strings be part of the name, for example: “Scream like a banshee”, then unfortunately the track name will be truncated to “Scream”.

This problem is solved by allowing the user to say “Quote” and “Unquote”, or “As Follows” to indicate the boundaries of the desired phrase. For example:

¹⁹ Dictation speech recognition allows arbitrary names to be recognized. If the recognized name contains the substrings “Like” or “As In”, then these substrings and everything after are simply removed, leaving only the desired name string. The reason they are included is to give more words to the N-Gram analysis function of the Speech Recognition Engine. The extra descriptive words can tip the balance in favor of the desired meaning of the intended word.

“Name this track *as follows*: Scream *like* a banshee” overrides the presence of “Like” and includes the entire phrase so indicated as the desired name.

Spelling it Out

SayPlay allows the user to spell out a name when it isn't recognized any other way. Spelling out a name is done using the following command phrase: “Name this track *spelled* *W., O., W.* (*or should I say “double-ewe, oh, double-ewe”*). The keyword “spelled” places the recognition engine into an alphanumeric spelling mode that accepts numbers and letters. Then the resulting recognized string, in this example: *W., O., W.*, is parsed to remove each period, convert to lower case, and concatenate all the letters into the word used for the name: “*wow*”. Spelling out a multiple-word phrase is not possible, since the “space” character is not recognized in the spelling mode of the Windows Speech Recognition. Hence, all recognized letters are concatenated into a single word. For a detailed step-by-step diagram of the functions, see: Figure 29 Activity Diagram for naming a track by spelling out the name.

Spelling out the name is good as a last resort to assigning a name, but becomes cumbersome if it is required for each subsequent command that refers to that

name. In the next section, we improve the accuracy of recognizing commands that refer to names once they have already been assigned.

Improving Accuracy in Referring to Assigned Names

Assigning a name correctly is the first piece of the puzzle. Having that name correctly recognized when referring to the named entity is the second puzzle piece. Just because the name has (finally) been assigned correctly, does not mean that it will always be recognized correctly in subsequent commands. Aside from the training-based performance improvement that results from normal use, there are other more direct ways of improving recognition of names: Adding each new name to the Speech Dictionary, preventing recognition of confused names, and loading the names into the grammar in the Speech Recognition engine.

Adding Names into the Windows Speech Recognition Dictionary

If a name is so difficult to recognize that the user must spell it out, or if the word is foreign, esoteric or is simply a made-up word, then it is possible to manipulate the “Speech Dictionary” directly, using techniques described in Appendix D: The Windows Speech Dictionary.

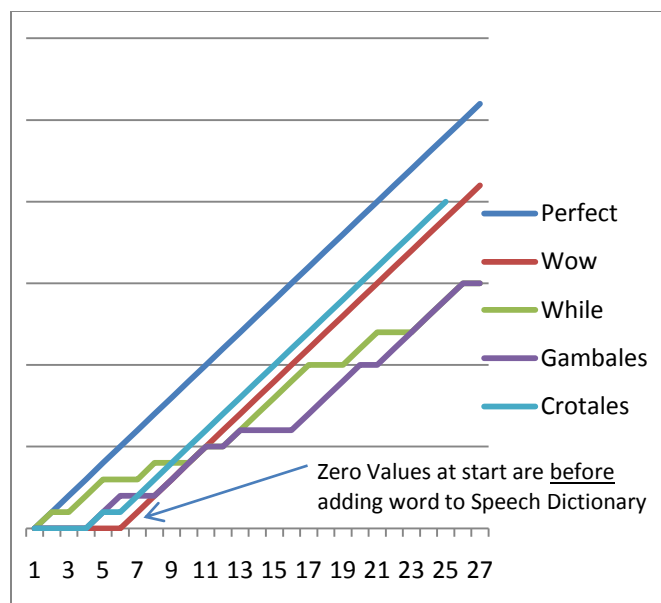


Figure 16 Cumulative Success Before/After Adding Name to Speech Dictionary

The test for the above is given in Table 11 Test 4 Test Procedure: Tricky Names.

The cumulative success of each track command is shown, before and after track names are added to the Speech Dictionary. Each name was assigned to a respective track, and then commands were issued by name on those tracks.

Then, after several attempts, the names were added to the Speech Dictionary, after which the cumulative successes improved, as follows: The name Wow was successful each time after it was added to the Speech Dictionary, Crotales failed once, but was successful in the remaining attempts. Gambales was still misrecognized occasionally, and this was very dependent upon pronunciation

Once track names are assigned, the user must be able to refer to those tracks by name. The same techniques used to initially acquire the name can also be used when referring to the name. These techniques were to elaborate using “Like” or “As In”, and also to spell out the name. However, this is somewhat unnatural, since one would expect the system to get better at recognizing the name after having gotten it successfully the first time. To address this, a technique is employed whereby all of the names of tracks are loaded into the Speech Recognition Engine grammar as possible target choices.

Loading Names as “Choices” in a grammar in the Recognition Engine

Intuitively, having a ready list of possible names should give the speech recognition engine the advantage of better recognition rates, but for some reason, certain names were difficult to recognize, even when they were included within the list of names loaded into the grammar.

The problem of having to continue to spell out a name even after having assigned it by spelling is solved by loading the name into the track command grammar, as one of the choices for the target of the command. See Figure 15 Grammar structure for issuing track commands by name: the Session Track Names at the

lower right of the diagram are those which are loaded into the grammar. Session Track Names are among the set of choices for the target of a track command. These are loaded into the grammar structure whenever the user issues the command to do so. Currently, that command is “Computer, please refresh the session”, but it could easily be “Computer, please update the session track names”, or some other intuitive command phrase. For a detailed diagram, see: Figure 7 Activity Diagram for loading track names into the loaded Grammar

When names are loaded into a grammar, each becomes one in a set of choices. As a result, recognition accuracy increases and it is no longer necessary to spell out the names. This is because the Windows Speech Recognition engine no longer has to rely on Dictation Speech Recognition to recognize the name. These claims are validated by the experimental results shown in Figure 17 Recognition Rates after Successive Improvements, and in Figure 21 Recognition Accuracy after loading homophones into loaded grammar. The grammar being referred to is the Track Command grammar, shown in Figure 15 Grammar structure for issuing track commands by name. The test procedure for the results below can be found using this link: [Table 12 Test 5: Effectiveness Adding Therein to Dictionary and Grammar.](#)

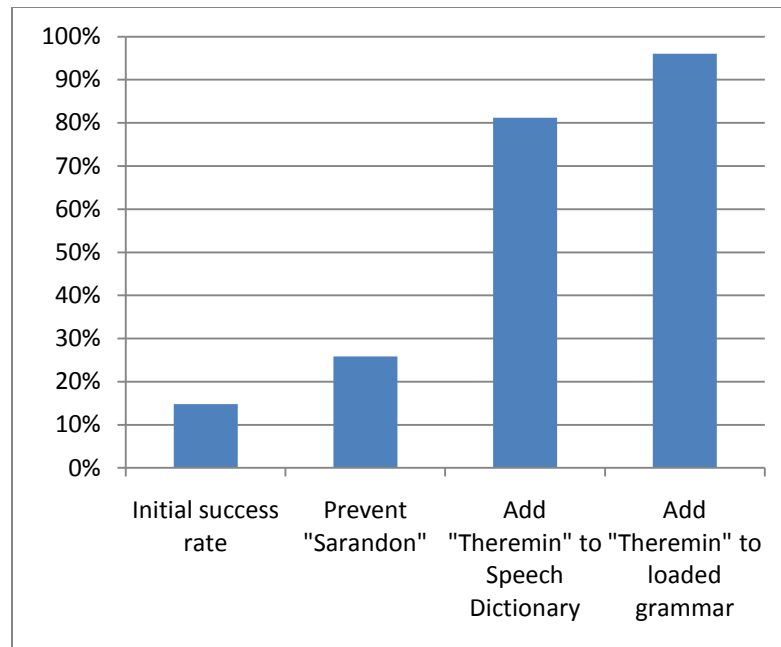


Figure 17 Recognition Rates after Successive Improvements

Discussion of results:

If “Theremin” was not initially in the dictionary, then how was it ever recognized correctly 4 times in the first 27 attempts (14.8%)? It must already have been in the Speech Dictionary, and the act of recording a pronunciation when adding it to the dictionary allowed the measured performance increase.

Once “Sarandon”, the primary source of confusion, was prevented from being recognized, accuracy improved to 15 of 58 attempts (25.8%). A leap in performance to 81.2% ($134/165 = 81.2\%$) occurred after “Theremin” was added

to the Windows Speech Dictionary. Please see Appendix D: The Windows Speech Dictionary for how to add and remove words to/from the speech dictionary. And another leap in performance to $(97/101 = 96\%)$ occurred after loading the name into the running grammar for track commands. See Figure 7 Activity Diagram for loading track names into the loaded Grammar.

Since there is reasonable suspicion that in addition to the actions taken to improve recognition, the sheer number of command repetitions plays some part in the improvements measured. Therefore, in order to eliminate the influence of machine learning on increased recognition performance, a follow-up experiment was conducted in which “Theremin” was first added to the Speech Dictionary (without first making so many command attempts). Then recognition success rates were measured before and after the name “Theremin” is loaded into the grammar. The results for two complete experiments are given below.

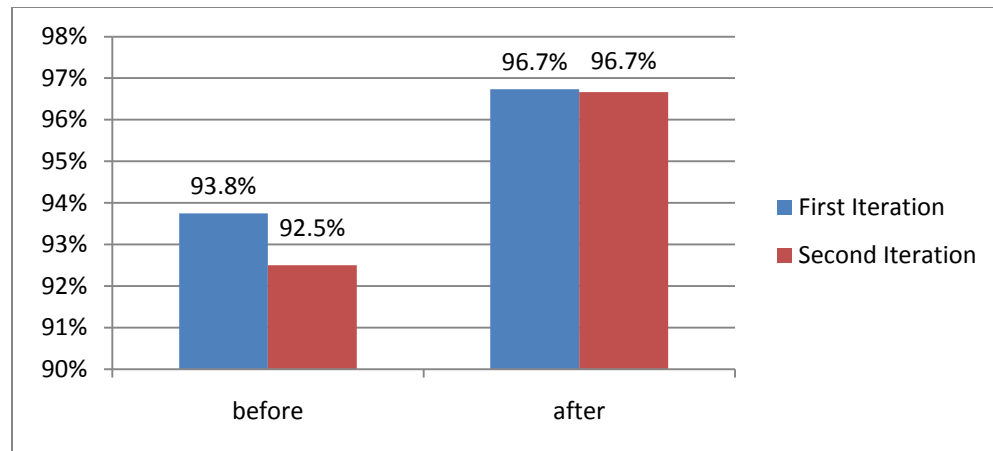


Figure 18 Recognition Rates Before/After Loading "Theremin" into Grammar

This experiment was further carried out on a larger set of tricky names, instead of only the name "Theremin" as shown in Table 4 Confused and Tricky Names.

Notes: Wow: Had to be spelled out to get it to work the first 7 times, even after adding it to the dictionary. This was because it was still confused with "While".

It didn't work without having to spell it until after it was re-loaded into the grammar. After that, it worked correctly 9 out of 9 times. The complete tally for

all tricky names is as follows:

Intended Name	Confused Name(s)	Elaboration/solution
Alto Sax	"alta sachs", "up to sax", "otto sax/sachs"	high saxophone
Bass	Base	Bass guitar
Cow bell	Cal bell	Dairy cow meadow sound
Crotales	Croat Olives	NA, had to Spell it out
Djembe		Must add to dictionary
Kazoo	Kazue	
Raisins	Reasons	dried grapes, Calif. Raisins

Sticklavier	N/A (it is a made-up word)	Must add to dictionary
Tapping	Taping	
Theremin	Sarandon, Thurman, Fairman, Salmon	Russian Inventor
Violins	Violence	Instrument
"Vocal Scat"	"Vocals get" "focal scat" "Vocal scout"	Jazz scat singing
Wow	While	NA, had to Spell it out

Table 4 Confused and Tricky Names

Word/ Technique	Initially Neither in grammar nor in dictionary	1 st load into Grammar In grammar, (not in dictionary)	Add to Dictionary (Not in grammar)	2 nd load into Grammar in grammar and dictionary	After restarting In grammar and dictionary (cont.)
Alto sax	=2/2	=3/3	=2/2	=2/2	=14/15
Bass	=1/5	=4/4	=1/5	=3/3	=14/14
Cowbell	=2/5	=5/5	=2/2	=2/2	=13/13
Crotales	=0/4	=9/21	=5/5	=2/2	=14/14
Djembe	=0/2	=3/15	=5/5	=4/4	=13/13
Kazoo	=2/2	=5/5	=2/3	=2/2	=13/13
Raisins	=1/11	=5/8	=0/3	=13/17	=16/18
Sticklavier	=0/4	=5/5	=0/1	=2/2	=13/13
Tapping	=2/2	=5/5	=2/2	=2/2	=13/13
Theremin	=2/2	=3/3	=2/4	=2/3	=13/14
Violins	=2/2	=5/5	=2/2	=2/2	=15/16
Vocal scat	=2/5	=3/3	=3/3	=2/2	=13/13
Wow	=0/11	=0/8	=5/5	=4/4	=15/15
Wow, spell it	N/A	=7/10	N/A	N/A	N/A
Notes:	x<59	59<x <162	163< x <205	x >205	

Table 5 Success/Attempt ratios for Adding Tricky Names to Dictionary and Grammar

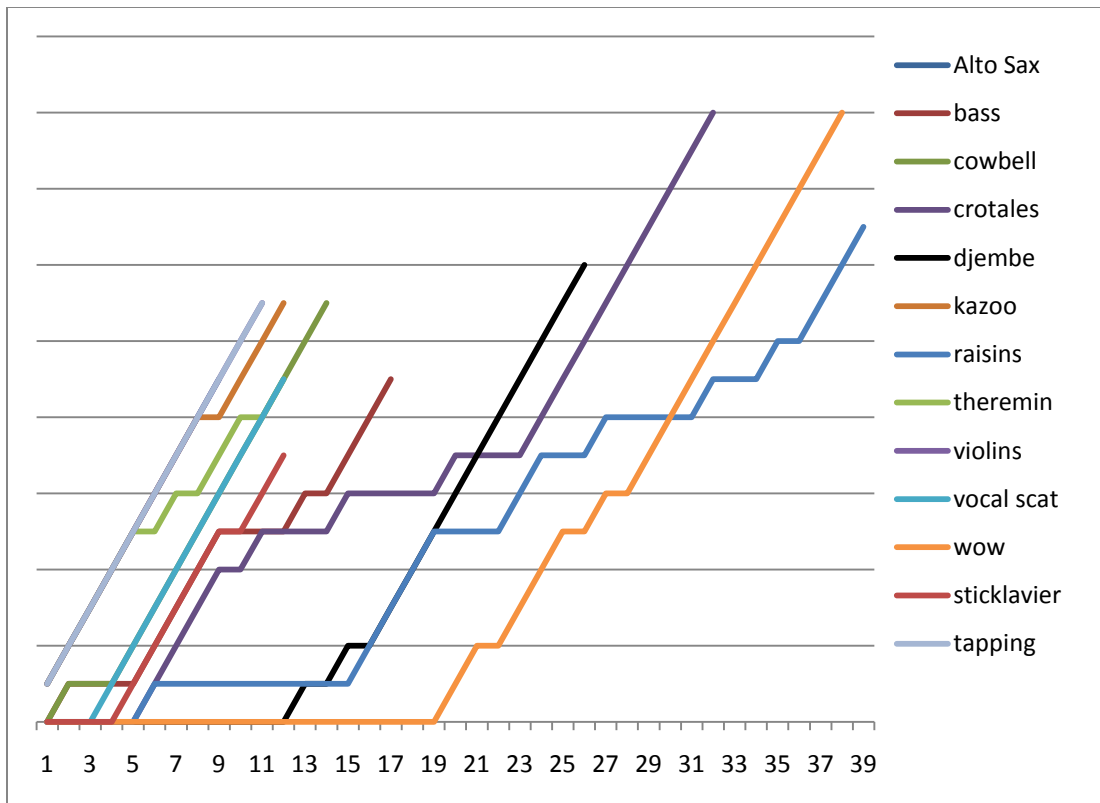


Figure 19 Cumulative Successes Before and After loading into loaded grammar

Discussion of results of Tricky Names and Dictionary/Grammar

Purpose: Expand beyond “Theremin” to include the many naming words that exhibited recognition problems. Goal is to determine whether or not they can be interpreted reliably.

Interpretation: The chart showing cumulative success shows a perfect string of successes as a steeply inclined smooth line. The key events of adding words to the dictionary and loading them into the grammar are indicated in a more detailed view of the graph, to illustrate the effect of these two improvements.

Conclusion: Loading names as choices into the grammar, so that dictation speech recognition is not required, does provide a measurable improvement in recognition accuracy, even for the trickiest words. Adding words into the Speech Dictionary also provides measurable improvement in recognition accuracy, especially when a word is foreign (Djembe) or made-up (Sticklavier).

Elaboration, and Loading Grammars using Homophone Pairs

Table 14 Test 7: Homophone Tests of Elaboration and Loading into Grammar describes a multiple step experiment to measure the effectiveness of Elaboration, and subsequently, the effect of loading names into the grammar. It uses Homophone pairs, which are words that sound the same but are spelled differently, like Presence and Presents. The only way for a speech recognition engine to correctly discern words that sound the same is through the context in which they are used. The way speech recognition gleans context is with

neighboring words. Thus, elaboration may allow one spelling to be recognized over the other.

The experiment uses pairs of homophones as input, first to see which spelling of the word is usually recognized. For example, “Name this track ‘BASS’” usually results in the name spelled “BASE”, by default. Next, attempts are made to use Elaboration to assign the other (non-default) spelling of the word (in this case “BASS”). If elaboration fails, then the name is assigned to the non-default spelling of the homophone by spelling it out. (Name this track spelled B-A-S-S). Next, several attempts are made to issue commands on the item named the non-default spelling of the word. For example, “Mute the Bass”, “Solo the track named Bass”. This establishes whether or not the unusual (non-default) spelling of the homophone continues to be misrecognized as the more commonly recognized (default) spelling. i.e. even after having assigned the name, it fails to distinguish the homophone in the unusual sense of the word. For example, if we were to name a track “Thyme”, which is the non-default spelling (“Time” is the more usual default spelling), then we can only hope that elaboration tips the balance toward the desired spelling of “Thyme”. Otherwise we must assume the system favors “Time” simply because it is less unusual.

Finally, this non-default spelling of the name (Thyme, in our example) is loaded into the track command grammar. Then more attempts to issue commands on this track are made. This shows that as a result of loading the name into the grammar, the unusual spelling of the homophonic word is finally reliably recognized over the more usual spelling of the word.

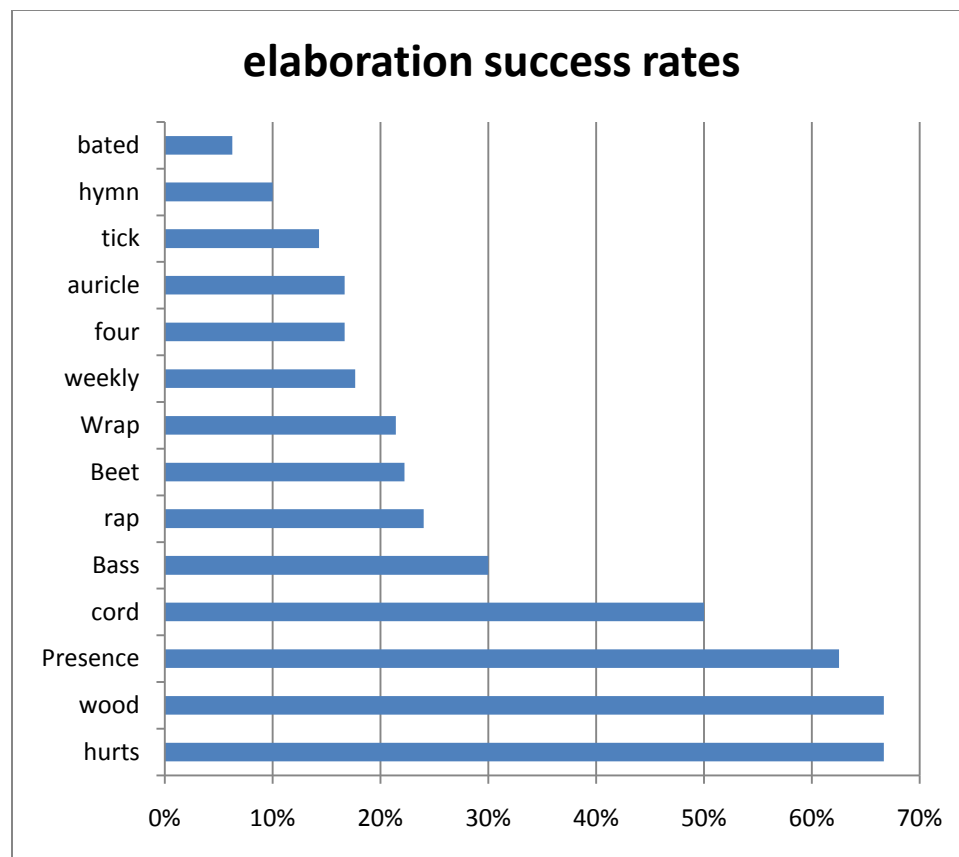


Figure 20 Recognition Accuracy when assigning names using Elaboration

The homophone pairs which found no successes in all attempts at using Elaboration were: Airy/Aerie, Aural/Oral, Away/Aweigh, Band/Banned, Basses/Basis, Belle/Bell, Bowed/Bode, Cymbals/Symbols, Haves/Halves, Hay/Hey, Him/Hymn, Loot/Lute, Lyre/Liar, Pairs/Pears, Paws/Pause, Rhyme/Rime, Root/Route, Sync/Sink, Thyme/Time, Tooter/Tutor, Undue/Undo, Weigh/Way, and Wrest/Rest.

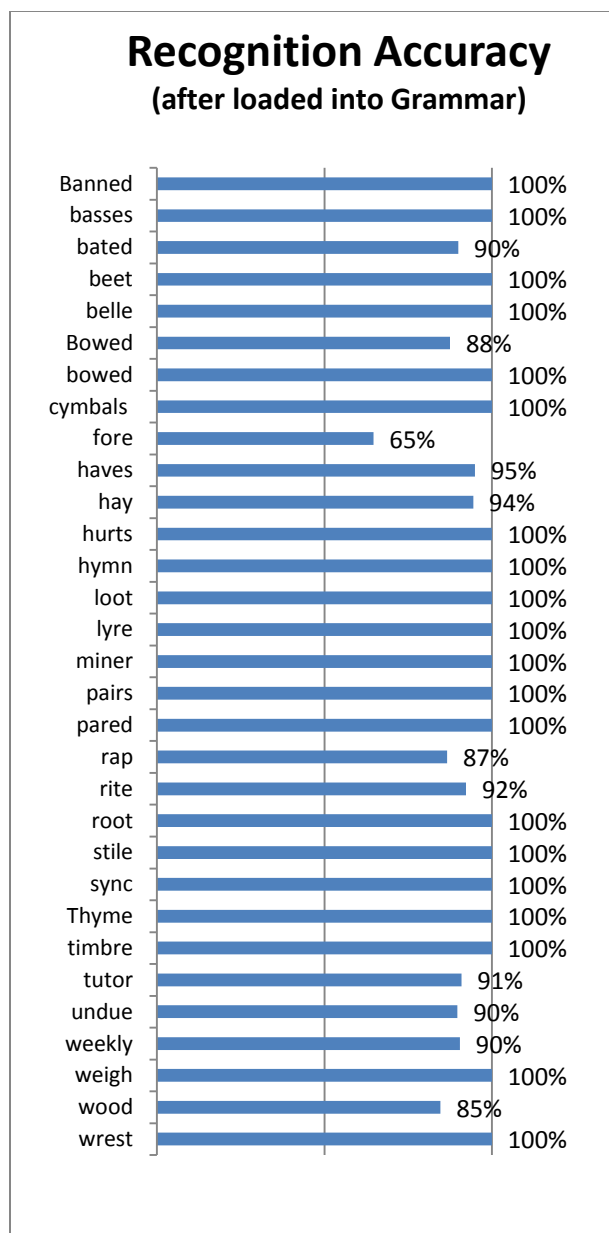


Figure 21 Recognition Accuracy after loading homophones into loaded grammar

The non-default homophone of the homophone pair is used as the name that is loaded into the grammar. The default homophone is more easily recognized using Dictation speech recognition, and therefore is not as good a test case. Using the homophone spelling which is decidedly harder to recognize contrasts the near-zero recognition accuracy before loading into grammar, with much-improved accuracy (near perfect for many) after loading into grammar. These non-default homophones had to be assigned to their respective tracks using the feature which allows one to spell out the desired track name.

Discussion of results:

Summary:

The experimental results show that a high level of recognition accuracy is achieved when names are loaded into the grammar as choices for being the object of the command action. This was true for almost all examples tested, once the non-default spelling of the homophone is loaded into the grammar as a choice of named items for the particular command family. However, when assigning the names, elaboration was found to be generally ineffective, while spelling out the name of the tracks worked well as a last resort.

Elaboration:

Elaboration was unsuccessful most of the time [for homophones]. In a few instances, it was strikingly clear that the additional words provided as elaboration

were enough to make the difference between one spelling of the homophone and the other. These cases are taken individually:

Hertz versus Hurts: The default result of naming was “Hertz”, probably because “Hertz” is a proper name already. When elaborating to say: “Name this track hurts like pain”, the result was successful 2 times out of 2. However, referencing the “hurts” track using elaboration was only successful 1 of 3 times. But, once the name “hurts” was loaded into the grammar, it was successfully referenced 15 of 15 times without elaboration.

Minor versus Miner: Minor was the default, but when adding “as in forty-niner” as elaboration, it returned “Miner” on 2 of 3 attempts. Although a subsequent reference failed, once the name “Miner” was loaded into the grammar as a name option, the recognition accuracy went up to all 11 of 11 attempts.

Presence versus Presents: Using the elaboration “Presence like in attendance” tipped the recognition engine toward “Presence” from the default result “Presents”. This was successful in 3 of 3 attempts. However, saying “Presents like gifts” failed to produce “Presents”, but rather produced the other spelling “Presence” of the homophone pair.

Wood versus Would: The default result was “would”. When elaborating to say: “Name this track wood, as in wooden”, the result was successful 2 out of 2 times, and after loading “wood” into the grammar, commands had a 20/24 success rate.

Discussion of anomalies:

Different default when singular from when it is plural:

“Cord” was recognized as a singular word, but when plural, the other spelling of the word: “Chords” was recognized. Elaboration, by saying “Like Piano Chords” “Like extension cord”, or Haves and halves, or Chorale and corrals.

Favoring Proper Nouns (even foreign words) over common words:

Occasionally the default name assigned by the track naming command appeared to be a foreign word. The three cases found were: “peres” for pairs, “shanti” for shanty, “wei” for weight. In addition, the names Orion and Ryan were assigned for “rhyme”, and Kazue for “kazoo”. (Earlier, several observances of “Alta Sachs” or “Otto Sax” appeared for the instrument “alto sax”. Each of these words is flagged by the Microsoft Word spell checker *when not capitalized*. If they are capitalized, then the spell checker accepts them. This was my clue to the following possible explanation. The fact that these words occur in a phrase that assigns a *name*, actually tips the decision toward a proper name and away from the common usage of the homophone in the test. It must be that the analysis of the

entire phrase, beginning with “Name this track...” creates a greater statistical likelihood that the intended name is to be a proper noun, rather than one of a number of parts of speech that the homophones represent.

Imperfect Homophones:

Timbre and Timber are not always pronounced the same, and so are not really homophones. Timbre was pronounced “Tam-ber”, in this experiment. Also, the imperfect homophone pair of Bases and Basis are pronounced slightly differently in the second syllable. Fort and Forte are pronounced distinctly differently, the first is one syllable, the second, Forte, is two syllables.

Shortened, Slang or Recently Coined Words:

Sync is short for synchronization, and shares a homophone with sink.

Mic is short for microphone (mic), and shares a homophone with Mike. Trying to name a track ‘MIC’ would likely name it “mike” since Mike *is* a name. Trying elaboration on mic, saying “like microphone” might work, but likely it would have to be spelled out. The question is whether it would work after loading into grammar (or does it need to be added to the Dictionary – note how the spell checker flags it both Mic and mic).

Clipped “B” after “ck”:

Two words had their beginning clipped: Beats and Band were both frequently misrecognized as “eats” and “and” respectively. Something about the sequence of phonemes where “**ck**” is followed by B”, is hard to recognize, as in “Name this track” to “**B**and...”.

Conclusions of Homophone Experiments:

The graph in Figure 21 Recognition Accuracy after loading homophones into loaded grammar, shows a very high (100% in many trials) accuracy in track command recognition rates after loading the track names into the grammar. This is true even when the non-default name is assigned to a track. However, elaboration did not work in most cases, and when it did appear to influence the outcome of assigning a track name, it was not 100% reliable. Spelling out a track name is a more sure method of assigning the desired spelling of a homophone. Combining spelling it out with the high reliability of loading the desired spelling of the homophone into the grammar, the results are convincing that even tricky names like those which sound the same as other words, can be used to refer to objects of action in recording.

Elaboration gives the user a sense that the system embodies some intelligence. It is strangely miraculous when after repeatedly failing to recognize and assign an obscure name, it suddenly recognizes the name given an extra bit of information. The problem is that Elaboration is implemented as a simple rule applied to the string of text which fills the name slot of a phrase. The overall command phrase is still that of assigning a name to something. Therefore the words recognized as names tend to actually be names, like “Wei” (for “way”), or “Peres” (for “pairs”), or “Orion”, or “Ryan” (for “rhyme”).

If the rule could decouple the context surrounding the Naming part of the command (“Name this track”), from the context surrounding the name and its associated elaboration (“rhyme, as in poetry”) then perhaps elaboration would perform better. Further experiments using the homophones that did work well with elaboration: “Presents/presence”, “Hertz/hurts”, “Wood/would”, might yield a new rule that widens the performance coverage.

CHAPTER 5: CONTRIBUTION SUMMARY AND FUTURE WORK

This thesis describes using voice commands for controlling the process of recording music. Contributions toward achieving this are as follows:

Contribution 1: A system supporting the multi-track audio recording workflow, which responds to voice commands is developed and measured for accuracy.

The prototype system demonstrates the usefulness and effectiveness of the proposed hands-free workflow, and allows measurements to be made. The system supports the measurement of accuracy by logging status for each speech command issued.

Contribution 2: Changes and improvements made to Audacity, an open source audio recorder and editor, to support track commands issued by name. These include the means of getting and setting the state of all tracks Mute or Solo settings. Also added to Audacity was the means for setting a track parameter given the name of the track to which to apply the change.

Naming tracks and then referring to them by name is explored as a way to improve system flexibility and usability.

Contribution 3: Analysis of experimental data gathered from several test subjects in using the system to perform basic recording tasks is made. Conclusions are

drawn of the effect of accent on the rate of performance improvement provided by the Speech Recognition Engine's machine learning capability.

Contribution 4: Development and analysis of the Elaboration technique to improve name recognition is made. More specifically, Elaboration is explored as a means of improving the recognition accuracy of tricky names such as "Theremin". While it worked only sporadically, it led to the homophone experiments which were useful test cases, and to the realization that the command phrase itself favors proper names over arbitrary common words, when assigning names to tracks.

Contribution 5: The technique of loading all names as choices into a grammar in the Speech Recognition Engine is explored as an improvement in performance of commands that refer to named tracks. When names are added to the grammar in this way, performance depends on Keyword speech recognition rather than having to continue to depend on dictation speech recognition, which, as shown in earlier experiments is not quite as reliable, as seen in the prevalence of the "Wrong Name" failure designation.

Contribution 6: Two features of Windows Speech Recognition are measured for improving recognition accuracy, namely Adding a word to the Speech Dictionary, and Preventing recognition of confused words. Measurement of the effectiveness of the system requires counting the number of failures in so many

attempts. Measuring accuracy before and after improvements are implemented shows the validity of improvements. Improved performance can be measured using the statistical approach given in [Sirota] “Minimum Sample Sizes for Attribute Superiority Comparisons”. Any changes made should result in a statistically significant improvement in the distribution of Success and Failure. While it is intended that existing Human Computer Interface techniques will be used in this application, it was hoped that some entirely new techniques might be explored. The concept of “Elaboration”, as in providing extra information to aid in recognition of tricky names, is explored as a possible improvement to the task of assigning names.

The improvement in recognition accuracy was measured for several users and it was found that in all cases recognition accuracy did improve, although at different rates for different speakers. Voices with accents required longer periods of use before reaching acceptable levels of accuracy.

Progress was made in providing flexibility in how commands are phrased, by providing grammars which allow various phrase structures for the same command family. For example, a user could say “Mute the guitar track”, “Mute the guitar”, or simply “Mute guitar”.

It was discovered early on that naming items and subsequently referring to them by those names was an important way of issuing voice commands, beyond the

simple set of keyword commands. Naming tracks and referring to tracks by name was described and test results were provided. Numerous challenging cases in the form of “tricky names” were given, and improvement techniques were developed and measured results of their effectiveness were described.

Discoveries of techniques to improve accuracy were made in the areas of naming tracks and subsequently referring to them by name.

Contribution 7: The appendices to this Thesis constitute an organized presentation of the workflow, the problem and solution, experiments and conclusions, and an annotated bibliography in an easily accessible format.

Future Work on SayPlay

The following features have imminent use in SayPlay recording session control.

Named Sections of Time:

An important feature that was not implemented is the ability to name sections of a recording, for selection and playback. For example, “Name the selected section “Bridge”, or “Chorus”, or “Recapitulation”. And thereafter allowing the user to cue playback or recording to any of these named sections. The reason this was not implemented is that the necessary changes to Audacity were a bit more extensive than time allowed.

Renaming Existing Commands:

Another short-term improvement to the system would address the challenges in remembering exactly what the command phrases are. The means for renaming commands could improve the ease of remembering the exact words for a given command. Allowing the user to create custom command phrases, in addition to existing commands, should make the system easier to use.

The paradigm for renaming a command is similar to naming a track or time section. A similar feature is implemented in [Gorniak & Roy], “Augmenting ...” wherein the user utters the desired command phrase to be used for a mouse click event. The grammar structure for enabling this command is shown below:

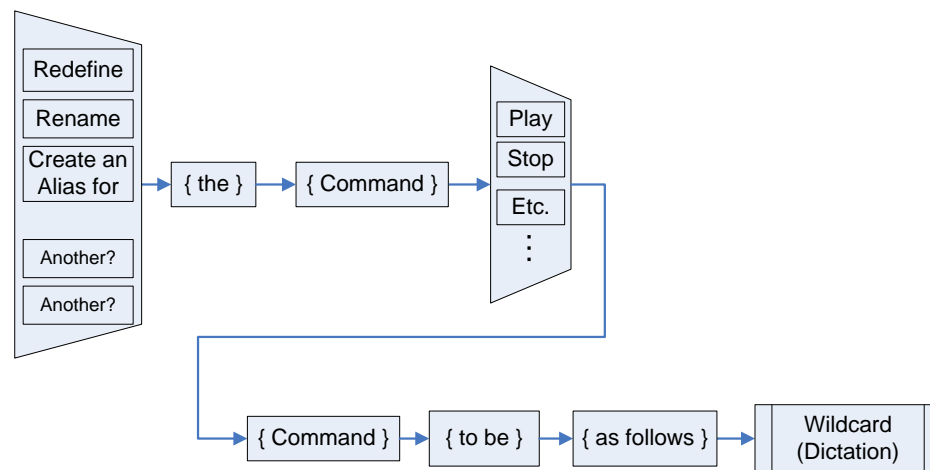


Figure 22 Grammar structure to rename an existing keyword command

Once the new command name has been assigned, the user must somehow confirm that it is correct. Only after confirmation will the new command phrase get added to the loaded grammar. Renamed commands, or command aliases can get loaded into the speech recognition engine using the same technique as used for adding track names into the loaded grammar. However, the new command alias will also have to be stored in a persistent fashion, so that from the moment of confirmation onward, and after quitting and restarting SayPlay, the new command phrase causes the desired command to take effect.

Possible Future Work:

The less immediate future direction could include some of the more ambitious ideas presented here.

Portable Device:

Beyond the convenience of having waveform displays of audio tracks, there is no reason why the recording and speech recognition software could not run on a tablet or even a smart phone. As long as the microphone/instrument inputs and headphone outputs can be furnished, the software could be made to run hands-free, once it is setup and ready.

Improved Event Logging:

Events could be logged in a format which is more easily analyzed. A set of tools for this analysis could be developed for analysis of user feedback (saying “Wrong” with a correction), for modeling the workflow (building up full-length tracks, then focusing in on specific sections for overdub, double-tracking, etc.), or for learning new commands or variations on existing commands. This work could feed into the use of Intelligent Agents and Latent Semantic Analysis.

Context Modeling:

A challenge of context modeling systems has been determining changes in the intention of the speaker. For example, whether a search is being refined, or whether a new search is beginning. In my approach, the user will state his intentions in a clear command, such as: “Computer, please enter Record Overdubs Mode”, or “Computer, please Audition for Mix”. However, these mode changing commands were rather arbitrary, and were not tested for effect. Discourse level processing will come primarily from these stated intentions. Stated intentions can have a wide ranging scope. A user may announce a long term goals with periodic effort being made, awaiting results from one stage to another, or a user may announce a more immediate goal. In either case, the computer needs to acknowledge the request and present the user with a list of

steps, sub-goals and tasks to accomplish on the way to meeting that goal. This serves both to confirm the correct understanding that the machine has of the goal, and to set the expectations of the user.

Intelligent Agents:

A set of software agents could carry out tasks on the recorded audio. Tasks such as tempo analysis, mistake detection, lyrics transcription, and take naming are all possible tasks. The essential tasks for the basic system are listening to voice commands, starting and stopping recording and playback, muting and soloing takes, setting volume levels, and naming takes and versions of songs.

Each of these tasks will carry a tally for mistakes made by the system, and a list of the commands that are mistakenly performed (or not performed). Improvement in performance can be measured by tracking the success and failure rates over time and in different recording scenarios.

Learning agents may make it possible for key aspects of system performance to improve over time. When a system is developed with “Agent Based” architecture, then every aspect of system performance is subject to improvement. [Russell]

The challenge of doing this within the Music Recording workflow described in this thesis is that all voice commands are channeled in one direction from the SayPlay program to the Audacity program. Success or failure of any given

command is indicated only by the user. Thinking of the system as a circuit of events, the event begins with the users verbal command, it is received and analyzed by the speech recognition engine, managed by SayPlay. An event with semantic information is directed to SayPlay by the speech recognition engine, and this event is analyzed by SayPlay which then generates and sends a command to Audacity which displays the command in the status bar, and carries it out if it can. Determining whether the command has correctly taken effect is done by the user.

Allow multiple track names in one command:

Allow the user to name an entity or set of them, so that it can be referred to by that name in subsequent commands. For example, “Name this track ‘Piano’”, followed by Play the track named ‘Piano’”.

Allow users to define new commands by speaking the command while manually performing the command (with mouse clicks, menus, etc.).

Create custom training sets consisting of texts that the users will read, while the ASR system trains on the keywords within the text. This text will be specific to the particular application, with command words that are appropriate to the task, such as recording music, or video, or lectures and presentations.

Creating Names and Referring to Named Items in Other Applications:

Can the techniques employed for voice command of music recording be transferred to other applications, such as video recording, meeting minutes

recording and summarization? Any interactive work flow that involves the assembly of arbitrarily created units, into components of larger assemblies could benefit from the “Naming things and subsequently referring to them by name” interactive paradigm. Tasks or sub-processes can be given names, and enacted by those names. Examples of this are:

Contact database entry and query

Defining (and naming/invoking) Recurring Routines

Describing the contents of a video or picture

Audio Information Retrieval based on voice query

Song selection and segment of song selection, by voice command. This feature is merely an extension of existing commands, their command grammars, and the corresponding commands within Audacity. New commands need to be added to the Audacity command set, to allow for label naming and recall by name.

New research into the detection of emotion in the voice of the speaker may be put to use, and the result would be a system that adapted appropriately. This goes further into context awareness.

And general Human Computer Interface for long term system planning, such as upgrade scheduling, (it is problematic when a computer is restarted overnight following system upgrade installs, but when the user has several documents and

windows open and ready to continue working on. The user must reopen all of them since the system has shut them all down, without any planning or scheduling).

Projects could be organized, background searches could be performed, and larger scale projects managed by the computer more actively and effectively.

What are the best Machine Learning techniques for expanding the range of commands available to the user?

Is it going to be effective to use Latent Semantic Analysis of all spoken commands, combined with prompts for clarification, to distinguish commands from user notes and comments? Will it be effective to divide the tasks of distinguishing which are commands and which are comments, from the tasks of clarification and discourse analysis, and to create agents that seek to accomplish these disparate goals? Will these agents conflict, or conspire against each other, or enter cycles where one undoes the work done by another, only to be redone by the first? Or will the agents become deadlocked and fail to continue because of some situation where each depends on the other, but neither can continue?

Conclusion

It became clear early on that simple translation of menu commands into voice commands would not be sufficient to allow smooth and natural control over the track settings for listening back to recorded takes during a recording session. Having to incrementally select up and down from track to track in order to select the desired one, or having to select them by number, are both impractical because they burden the user with remembering the exact mapping of tracks. This situation rendered the name as the best reference for working with tracks.

Overall, speech recognition requires patience in the user, because a period of training is necessary for the recognition engine to model the individual user's specific way of speaking. This period is also important for the user to become comfortable with the specific phrases for each command. Some successful events are necessary for the user to be able to adjust accordingly. Initially, it is best to use the standard set of track names {piano, vocal, guitar, drums}, which are loaded into the grammar from the start, in order to experience some initial success.

Name referencing commands require the use of Dictation Speech Recognition to assign names of a wildcard nature. It was found that track naming commands had lower recognition accuracy than keyword commands, due to the greater

difficulty recognizing tricky names relying upon dictation speech recognition.

Development efforts to improve the assignment of and reference to named entities showed measurable improvement. These developments were: spelling out the name, followed by loading all track names into the running grammar.

In addition, the features provided in Windows Speech Recognition for adding words to the dictionary, and for preventing dictation of confused words, are also available, and the resulting performance increase was measured. Finally, the techniques of Elaboration and the associated overriding Quotation technique were found to be less effective than imagined, but experiments with homophones revealed that the name assignment command favors proper names over similar sounding words which do not usually serve as names.

The specific domain of audio recording can benefit from voice commands, when an integral part of the workflow allows tracks to be named, so that they may be referred to by name.

The techniques used within the domain of music recording could be applied in other domains where the workflow involves the need for hands free control, such as camera operation and robot control.

APPENDIX A: STORYBOARD DESCRIPTIONS OF RECORDING
SESSION WORKFLOW

Recording a set of new pieces (also known as recording Basic Tracks)

Recording overdubs (additional layers on top of existing recordings)

Re-Recording sections of a piece, for later editing (punch-in)

Basic Preparation

Set Recording Levels, ensure each instrument microphone is functioning, has proper settings and placement, and that the signal path into the recorder is functioning and clean. Ensure proper signal levels are set by having the performer play at the loud extreme of their dynamic range, to avoid saturation during the peaks, and have them play softly to ensure a good signal to noise level. Once the recording equipment is all set up and working, the headphone monitor mixes need to be set to comfortable listening levels for the producer and each musician to hear the other instruments.

The musicians can warm up during the process of setting monitor mixing levels, and finally, they should make any adjustments to the tuning and intonation of their instruments.

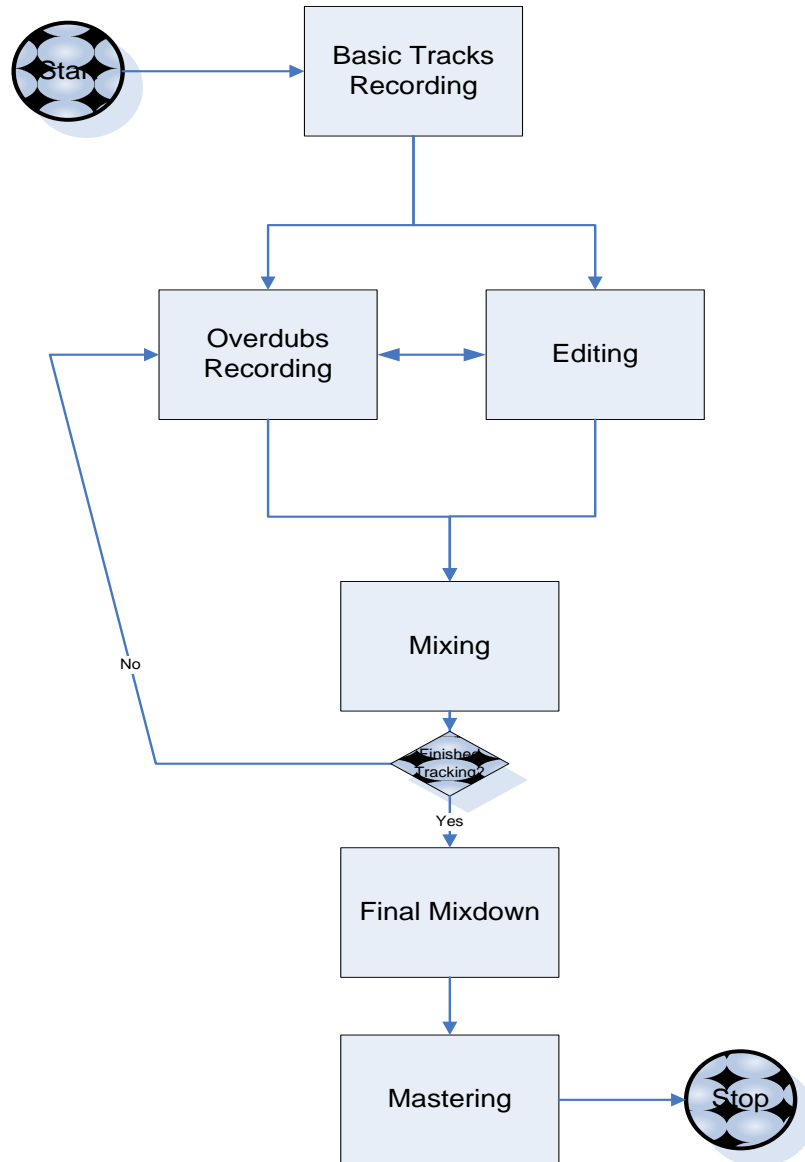


Figure 23 Recording Overdubs

Announce Basic Track

The producer will say a few words about the ordering of the recordings to be made, and more specifically about the first piece to record. These comments should be recorded by the system, as recording session notes. The performers may also offer insights about their interpretation, which may be of interest later on. So, all these conversations are recorded as notes for the session.

Once the producer announces the ready for recording the first take, notes are saved with the piece and take.

Recording the First Take:

The producer may ask the musicians to play the piece once without recording. It is helpful to record this take anyway, since often times it has none of the pressure of the first “real” recording, and it can sometimes turn out to be the best take.

Musicians are not discouraged from continuing if a mistake is made, and they can get a fresh perspective on the piece as performed in the studio surroundings.

Once real takes are recorded in earnest, performance mistakes can become more frustrating, sometimes resulting in even more mistakes and frustration.

One of the goals of the system is to remove the tension of the recording process, so that performers barely notice they’re being recorded, except when listening back to a recorded take. Imagine a recording session in which no recording is

going on. This is called a rehearsal. It may not be so different from one in which everything is being recorded. A rehearsal where everything is recorded captures a live performance but without the audience. This may be a picture of the ideal basic track. But, there are some challenges associated with getting everything recorded, so that it's easy to listen back to during the session and easy to access later on. This is where voice commands come to the rescue.

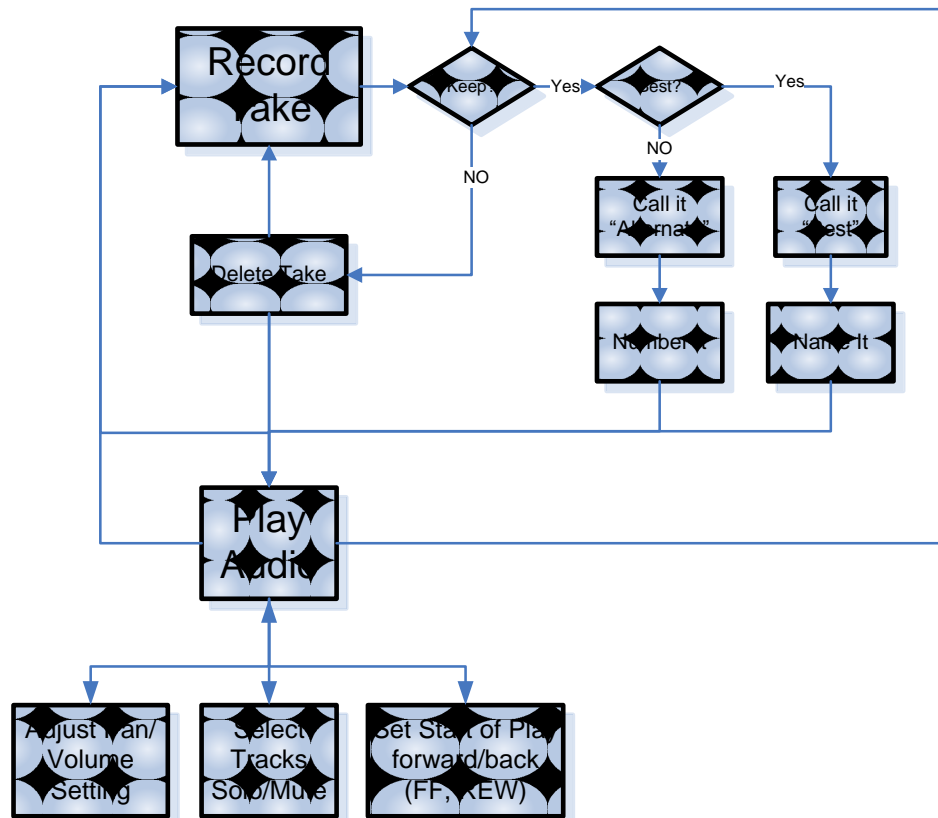


Figure 24 Workflow for recording Overdubs

The following table gives a storyboard for recording the first take of a performance:

Producer Says	System Does
Let's Start Recording or Another Recording	Starts recording, marks take number Start time is set on level exceeding a threshold
Let's Record a take or Record another take	Begin recording while playing back other tracks
Let's hear that back	Plays back last recording from start
Let's hear the first one played back	Solo the first track and start playback
From the last verse	Set play pointer to desired location, and play
No, the verse before	Set play pointer to desired location, and play
No, 4 bars later	Set play pointer to desired location, and play
No, the take before	Set play pointer to desired location, and play
Ok, stop playback	Stop playback
Everyone ready?	nothing
Let's work on the [cellos, vocals, etc.] tracks	Keeps tracks armed for recording

Re-Recording, the second, third, fourth takes, and so on...

Sometimes a producer may want several versions of a piece, played at different tempos, perhaps in a different key, or perhaps with different moods or intensity.

Or a producer may be focused on a very specific rendition, and will keep the band trying to reproduce it time after failing time.

The following table gives a list of terms and what they refer to. This is a starting point for the system to begin to understand what the producer means when asking for each particular item. There may be synonyms for these items, and

since all synonyms cannot be listed (predicted) in advance, the system can be taught what new ones mean, in terms of the ones in this table:

Term	What refers to
This take	The upcoming (yet to be recorded) take OR The take just recorded
Last take	The take just recorded
Next take	The take following the one commonly referred to. If the commonly referred to take is several previous, then next is the one following. If the commonly referred to take is the last take, then next is the same as This take.
(Un) Solo {This, named} Track	Mute all other tracks, to audition only the indicated track
Un/Mute {This, named} Track	Mutes/Unmute the named track

Table 6 Terms referring to track commands

“Fix it in the Mix”

Decisions are made as whether or not to record another take, based essentially on whether the last take was reparable. Digital audio recording software can provide a variety of tools for correcting problems in any recording, and so it is the producers decision based on what he knows can be done. For example, a few notes being early or late, just not quite right on the beat, can be edited so that they fall right on the beat as desired. This problem does not require re-recording. Adding titles, comments and notes to recordings: Frequently notations about the recorded version must be made, usually as indicators for further work to be done

or decisions to be made. Markers and labels are useful for doing this when they can be easily recalled by name.

Recording Music:

Music Recording is the process of recording an ensemble of musicians and then mixing them together into a pleasing blend. Each part of the ensemble may be recorded several times in order to get the best performance, and from each of these several recordings, shorter time spans can be edited together to create a seamless realistic performance. The process of obtaining this final edited performance is by repeated recording and listening back to the recorded parts to select which parts to use in the assembly of the complete whole.

The producer will ask the performer to play in certain ways, to make changes, or an individual performer/producer will ask to hear back a particular take, or will announce the name of a new take, and some distinguishing feature, to be recorded as a note (stored as metadata) about the take.

Recording Voice:

Usually called “Voice-Over” recording, the recording of an individual’s speech for dramatic, journalistic, or documentary purposes, is a process much like recording music, except that instead of synchronizing the performance with those of other musicians, the speaker (called the “talent”), must sometimes synchronize their performance with a video recording.

Training the computer on the prosodic components of a dramatic delivery, where the script is given, and various expressive “angles” are explored by the acting talent. Each “angle” has to be given a name, so the system can associate the prosodic queues with the differences in this dramatic delivery from others with the same, and with different “angles”.

Naming Events	System Does
Name This Track ...X	Assigns the name of the currently selected track to the name given by user
Name the recorded track...X	Assigns the name of the last (bottom) track to the name given by user Most recently recorded tracks are placed on the bottom.
Name this Take... X	Assigns the name to the current Take, as recorded by the context of the system.

Table 7 Naming Tracks and Takes

APPENDIX B: THE DEMONSTRATION SYSTEM

The demonstration system consists of a Dell Inspiron 1525 laptop computer, running the Windows Vista or Windows 7 operating system, and two software applications developed for this research, and a USB headset microphone.

The first application is a modified version of Audacity, an open source digital audio recording and editing program, available at:

<http://audacity.sourceforge.net/>

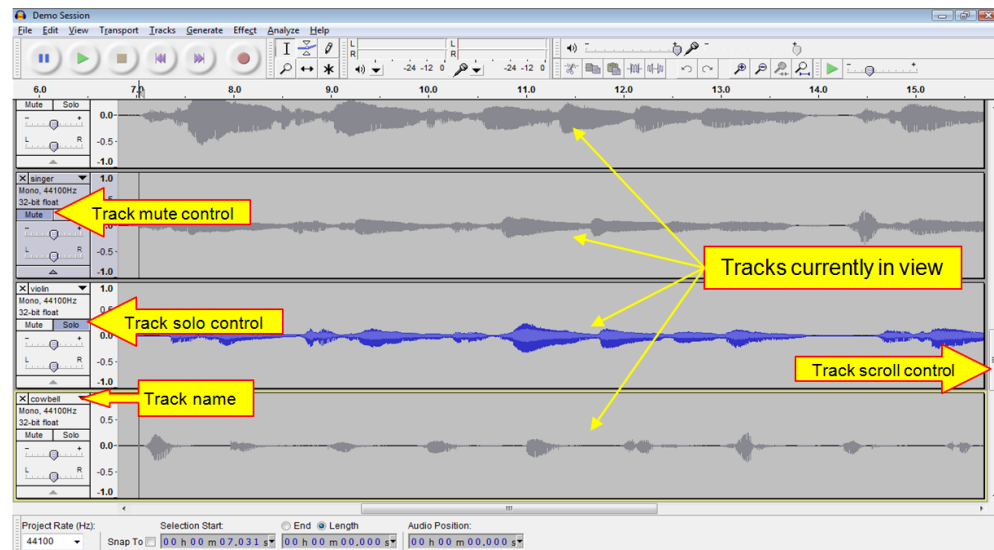
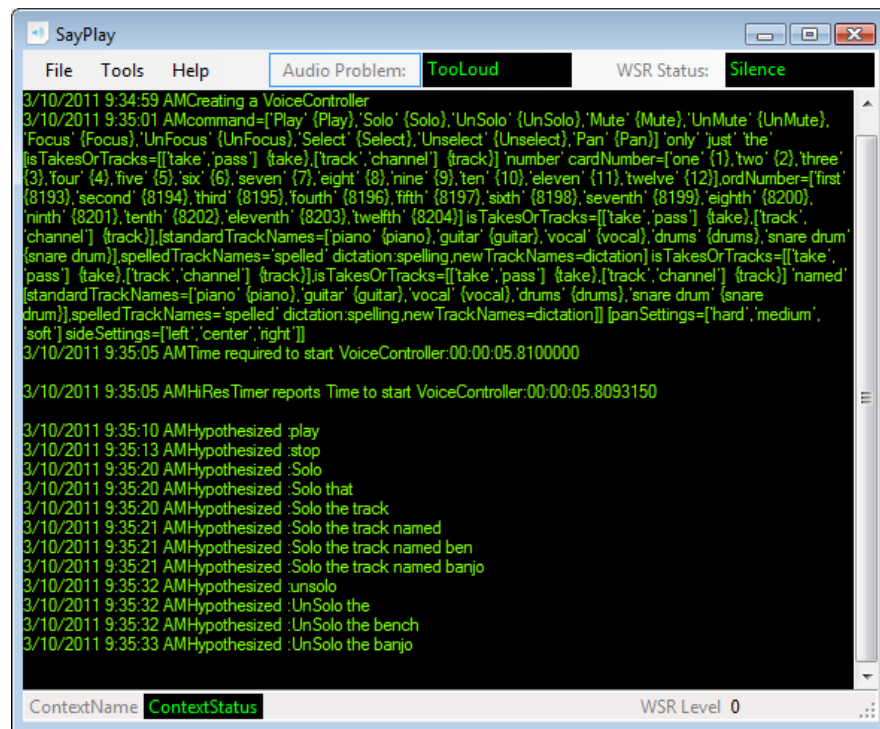


Figure 25 Audacity User Interface with annotations

The second program is called “SayPlay”, developed specifically for this research. SayPlay uses the Windows Speech Recognition engine to interpret speech recognition events from speech input, and sends commands to Audacity.



```

SayPlay
File Tools Help Audio Problem: TooLoud WSR Status: Silence
3/10/2011 9:34:59 AMCreating a VoiceController
3/10/2011 9:35:01 AMcommand=["Play" {Play}, 'Solo' {Solo}, 'UnSolo' {UnSolo}, 'Mute' {Mute}, 'UnMute' {UnMute},
'Focus' {Focus}, 'UnFocus' {UnFocus}, 'Select' {Select}, 'Unselect' {Unselect}, 'Pan' {Pan}] 'only' 'just' 'the'
'isTakesOrTracks=[[ 'take', 'pass' ] {take}, {track', 'channel' } {track}] 'number' 'cardNumber'=[ 'one' {1}, 'two' {2}, 'three'
{3}, 'four' {4}, 'five' {5}, 'six' {6}, 'seven' {7}, 'eight' {8}, 'nine' {9}, 'ten' {10}, 'eleven' {11}, 'twelve' {12}], 'ordNumber'=[ 'first'
{8193}, 'second' {8194}, 'third' {8195}, 'fourth' {8196}, 'fifth' {8197}, 'sixth' {8198}, 'seventh' {8199}, 'eighth' {8200},
'ninth' {8201}, 'tenth' {8202}, 'eleventh' {8203}, 'twelfth' {8204}] 'isTakesOrTracks=[[ 'take', 'pass' ] {take}, {track',
'channel' } {track}], 'standardTrackNames'=[ 'piano' {piano}, 'guitar' {guitar}, 'vocal' {vocal}, 'drums' {drums}, 'snare drum'
{snare drum}], 'spelledTrackNames'='spelled' 'dictation:spelling.newTrackNames=dictation] 'isTakesOrTracks=[[ 'take',
'pass' ] {take}, {track', 'channel' } {track}], 'isTakesOrTracks=[[ 'take', 'pass' ] {take}, {track', 'channel' } {track}] 'named'
[standardTrackNames=[ 'piano' {piano}, 'guitar' {guitar}, 'vocal' {vocal}, 'drums' {drums}, 'snare drum' {snare
drum}], 'spelledTrackNames'='spelled' 'dictation:spelling.newTrackNames=dictation]] [panSettings=[ 'hard', 'medium',
'soft' ] sideSettings=[ 'left', 'center', 'right' ]]
3/10/2011 9:35:05 AMTime required to start VoiceController:00:00:05.8100000
3/10/2011 9:35:05 AMHResTimer reports Time to start VoiceController:00:00:05.8093150
3/10/2011 9:35:10 AMHypothesized :play
3/10/2011 9:35:13 AMHypothesized :stop
3/10/2011 9:35:20 AMHypothesized :Solo
3/10/2011 9:35:20 AMHypothesized :Solo that
3/10/2011 9:35:20 AMHypothesized :Solo the track
3/10/2011 9:35:21 AMHypothesized :Solo the track named
3/10/2011 9:35:21 AMHypothesized :Solo the track named ben
3/10/2011 9:35:21 AMHypothesized :Solo the track named banjo
3/10/2011 9:35:32 AMHypothesized :unsolo
3/10/2011 9:35:32 AMHypothesized :UnSolo the
3/10/2011 9:35:32 AMHypothesized :UnSolo the bench
3/10/2011 9:35:33 AMHypothesized :UnSolo the banjo
ContextName: ContextStatus WSR Level 0

```

Figure 26 The SayPlay program written for this research

The “SayPlay”, application, and the modifications to Audacity are the primary focus of the development part of this Thesis.

A context diagram of the Audacity and SayPlay applications is shown below:

From an executive level, a program for managing the interaction with the speech recognition engine, and the events it generates.

A class hierarchy called a `CommandFamily` is used to make Voice Command events appear as a uniform type of event, and to allow special case handling to be done based on the individual type of command.

The primary `CommandFamily` methods are `BuildGrammarFromScratch`, and `HandleCommand`. These two methods are called at the highest level, for all `CommandFamily` objects, first to build the grammar to load into the Speech Recognition Engine, and thereafter, for handling the events generated by the recognition engine for each. So, all the specifics about grammar structure and special handling instructions are encapsulated into the subclass.

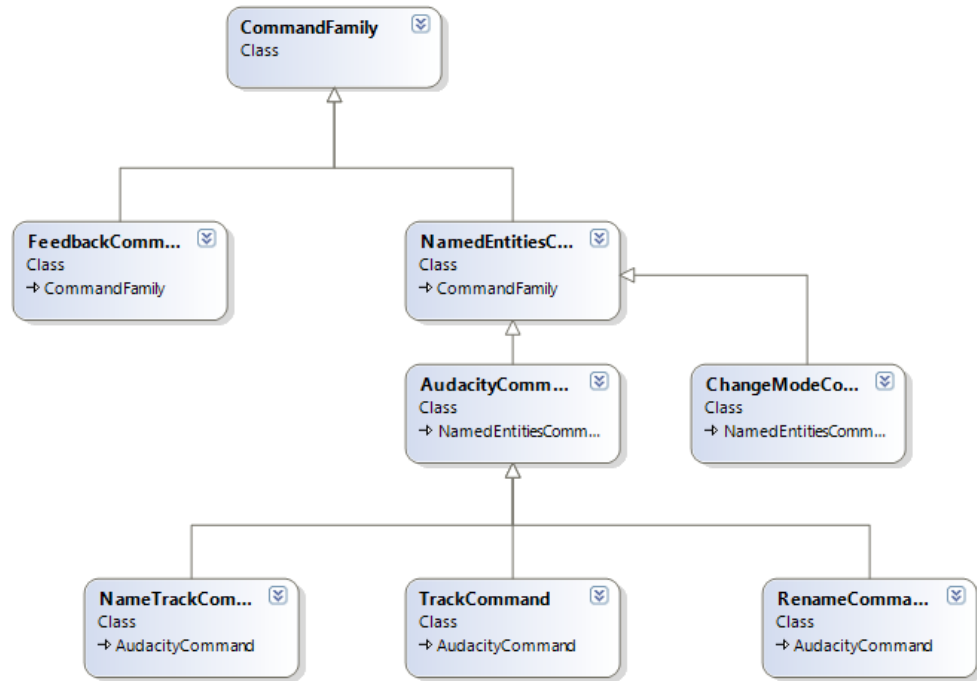


Figure 27 Class Diagram of the CommandFamily in SayPlay

A Context Manager is implemented to manage state, and to filter events through the current context (or state) of the recording process. This can allow the system to refine the semantic interpretation of certain voice commands. For example, a command that refers to ordinal numbered tracks, such as “mute the second track”, or “mute the second guitar take”, where the semantic difference is between “Track” and “Take”. “Track” is an absolute number, whereas “Take” indicates an offset from the first among the recorded guitar takes.

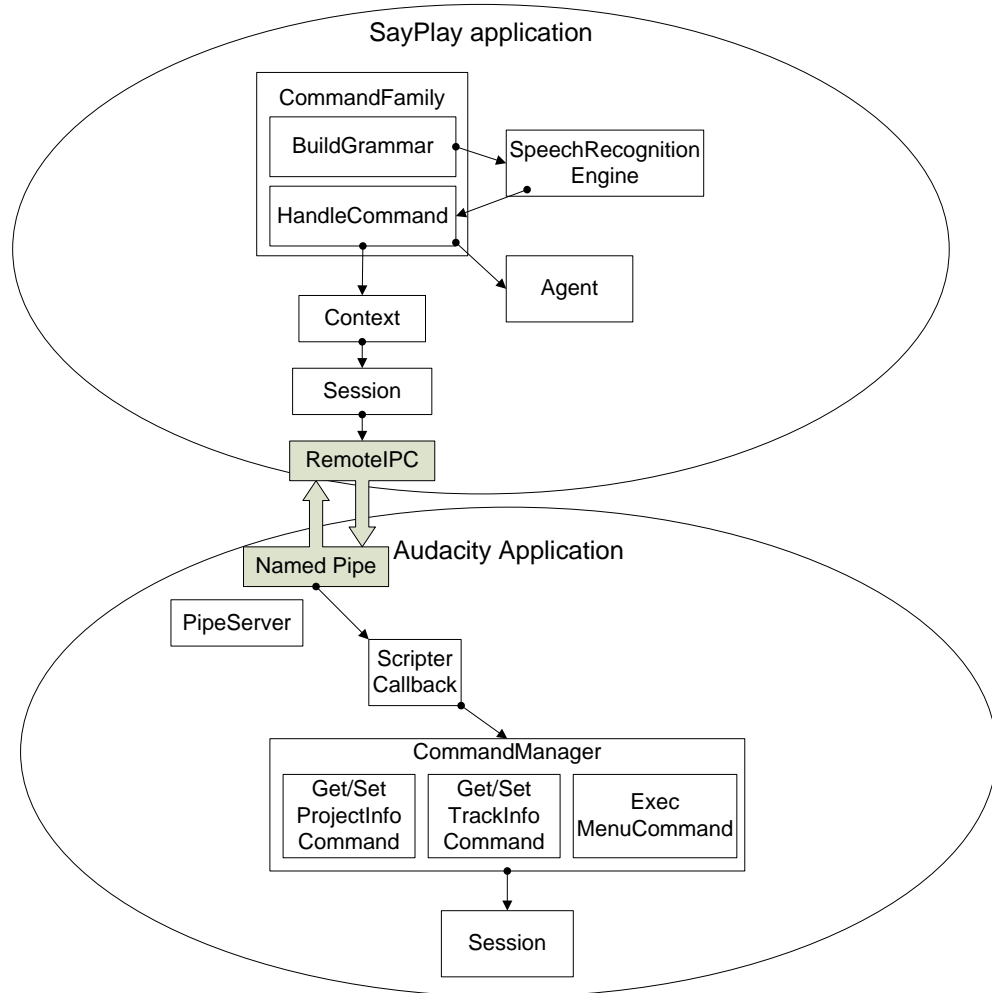


Figure 28 Context Diagram of SayPlay and Audacity applications

Voice commands originate from SayPlay and are sent to Audacity over a named pipe to Audacity, which returns status information to SayPlay.

SayPlay is the tool which accepts spoken commands such as “Play”, “Record”, “Pause”, and “Rewind”, and issues the corresponding command to Audacity.

These commands are the simplest level of control, and alone may not prove worthwhile, since the user is likely to need to use the mouse for entering text.

The following activity diagrams show the steps in the following actions: Naming a track by spelling it out, Naming a track by elaboration, and loading the names into the loaded grammar.

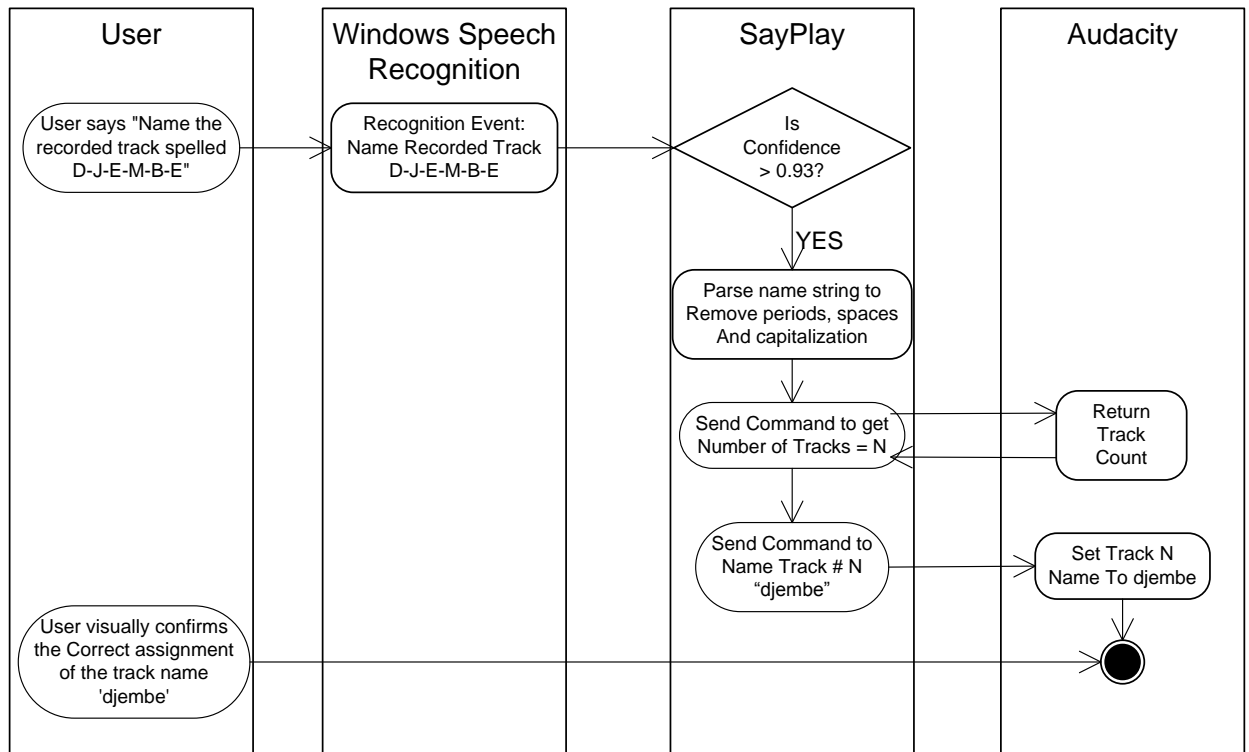


Figure 29 Activity Diagram for naming a track by spelling out the name

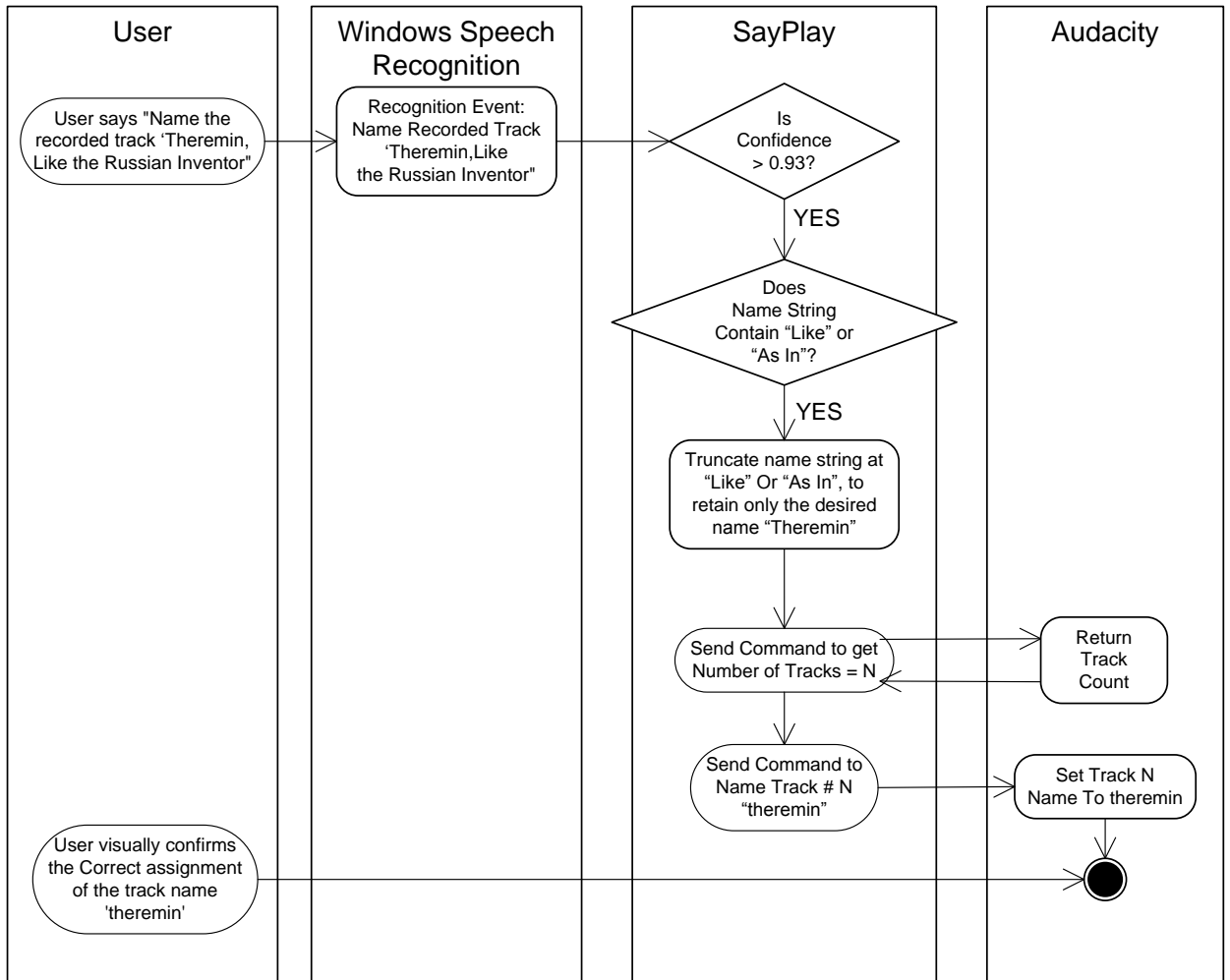


Figure 30 Activity Diagram for naming a track using "Elaboration"

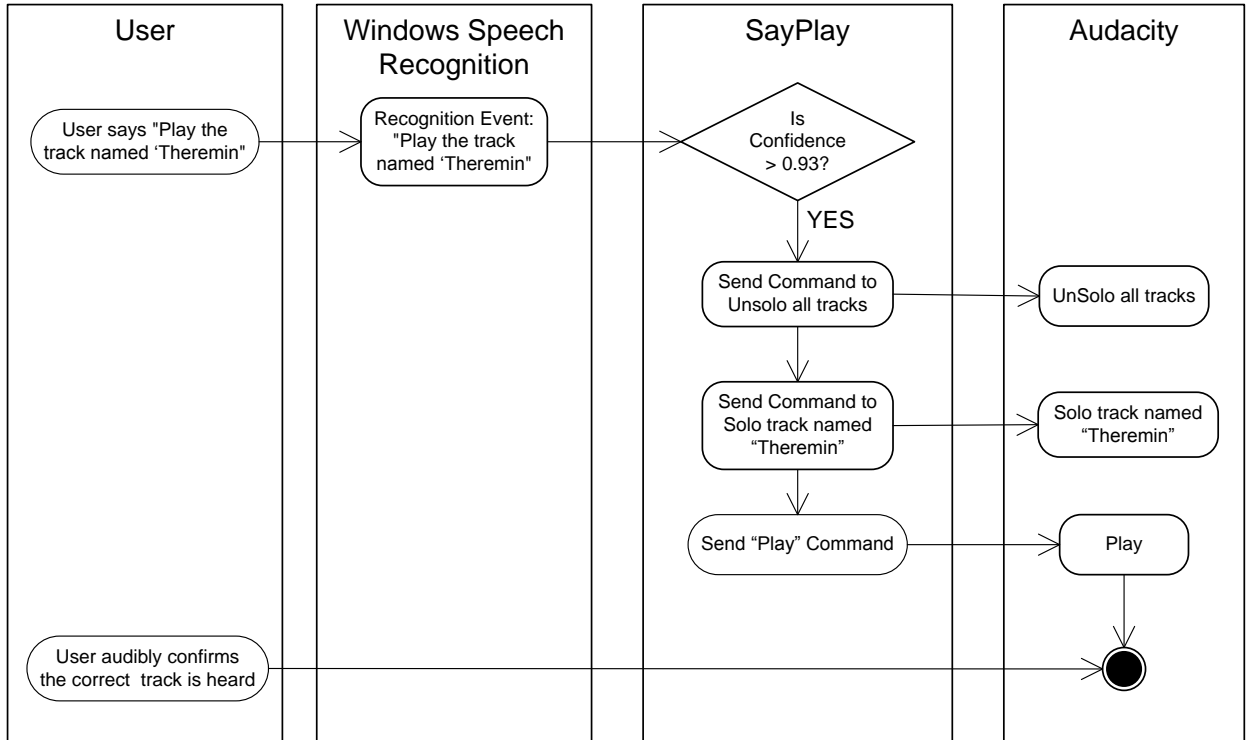


Figure 31 Activity Diagram for the Track Command to Play a Named Track

Finally, an Agent interface is provided, such that a list of agents can be registered with the event handlers of the Chain of Responsibility. These agents have access to the voice command data at the time of the detection of a recognized event, and can log recognition event information to a file, for later semantic and statistical analysis.

APPENDIX C: SETTING UP WINDOWS SPEECH RECOGNITION

Purpose: To measure the accuracy (as ratio of success to the total of success and failure) of the system at performing speech recognition commands. Various commands are exercised.

System Requirements: Windows Vista Business or Windows 7. Speech Recognition SAPI 3.5 or later, Visual Studio 2008, and the 2 projects: Audacity.sln and SayPlay.sln.

Software Under Test: “SayPlay” is the program which accepts voice commands and “Audacity” is the audio recorder and editing software used to record and play tracks of audio.

Setup: In preparation for using Microsoft Windows Speech Recognition, please open the Speech Recognition Options control panel, and select the “Set up microphone” configuration option. You must use a headset microphone for best results. These selections are indicated in the figure below.

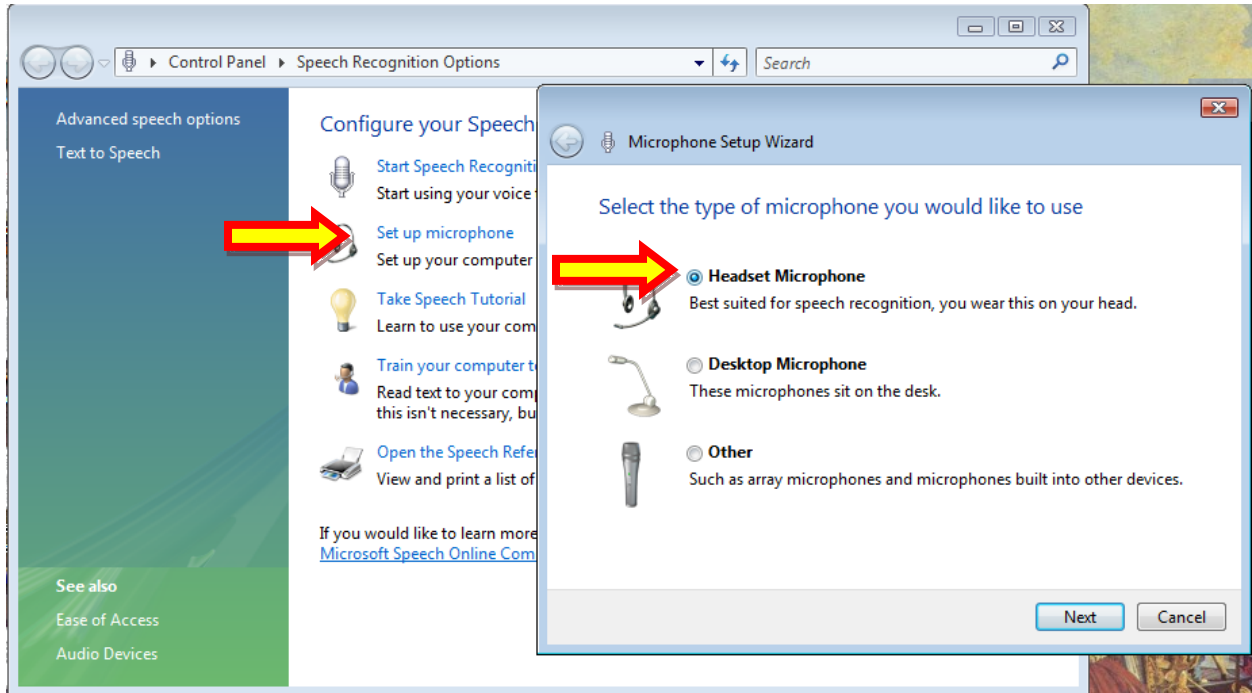


Figure 32 Speech Recognition microphone setup

The following dialog appears, allowing you to speak for awhile so the mic level can be optimally set.

As instructed, please heed the following advice:

Proper Microphone placement

Position the microphone about an inch from your mouth, off to one side

Do not breathe directly into the microphone

Make sure the mute button is not set to mute

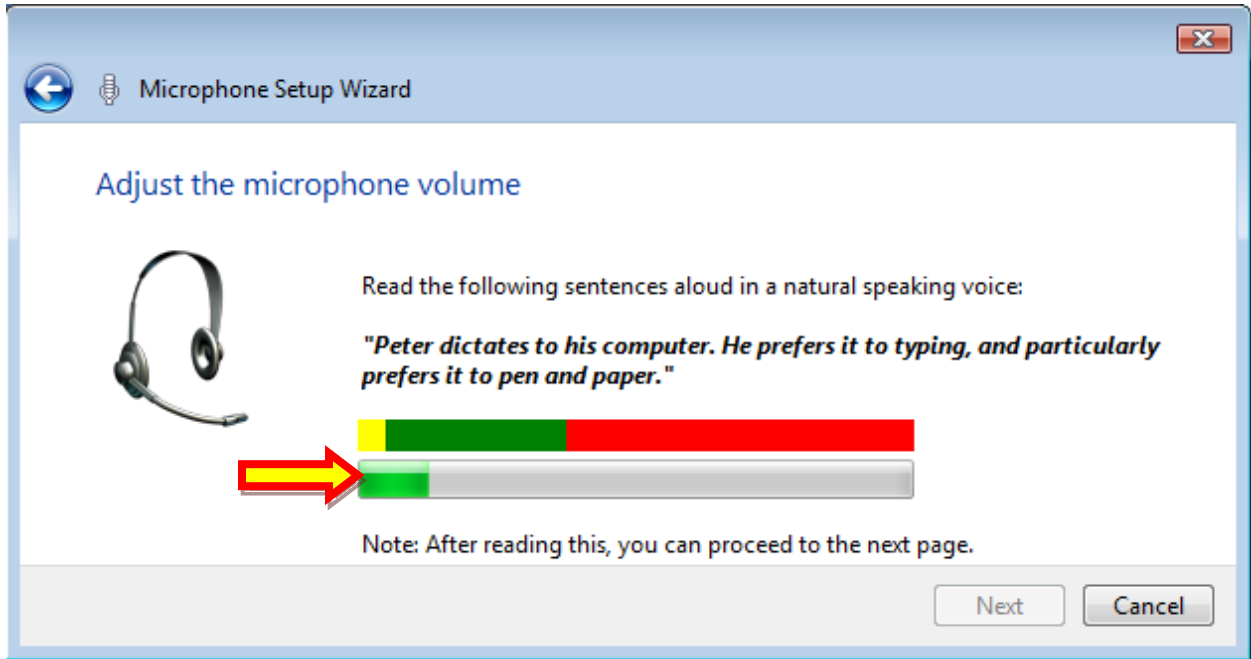


Figure 33 Speech Recognition Microphone Setup for levels

As you speak, the green bar, indicated above by the red arrow, will show the audio signal level. Setting the microphone level is done automatically based on this process, so it is important to speak at the same loudness level that you will normally use. If you speak louder than normally, the mic level will be set lower than it should, and, conversely, if you speak quieter, then the mic level will be set too high.

Once this process is complete, the next step is to set up a User Profile.

Enter the Advanced speech options, as shown below:

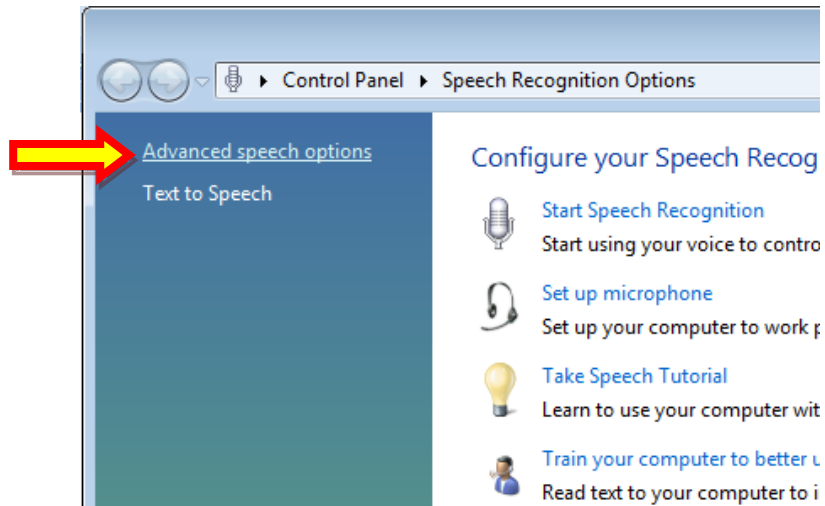


Figure 34 Speech Recognition Control Panel Advanced Options

Create a profile for the test subject, using the “New Profile” button, as shown on the window below.

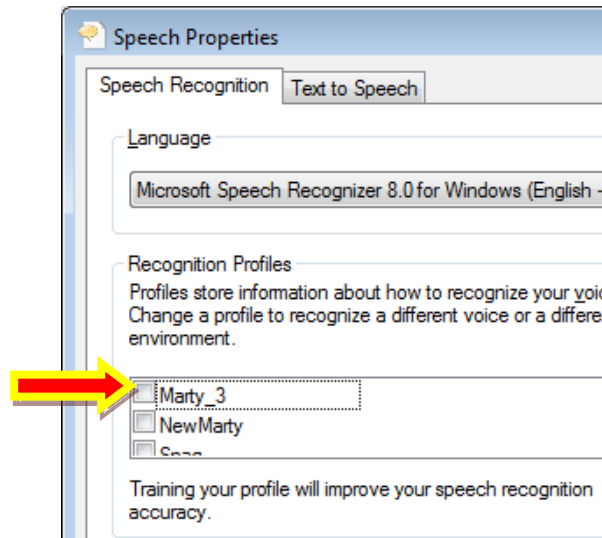


Figure 35 Speech Recognition profiles

It can help to verify that the microphone input signal level is a reasonable setting:

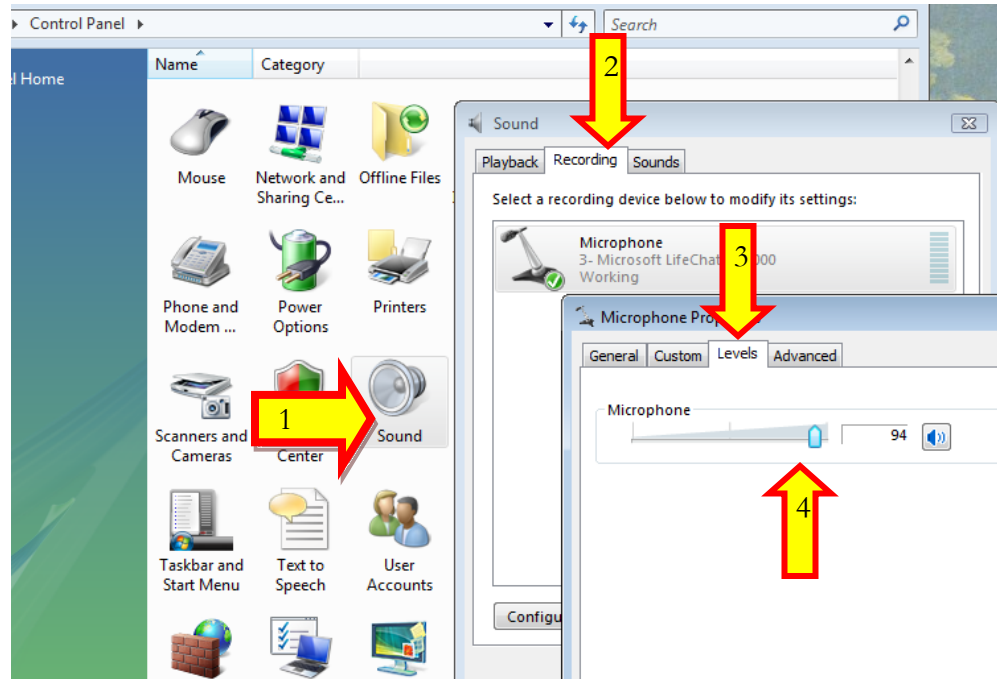


Figure 36 Selecting the Speech Recognition Control Panel for Microphone Level

Next, perform one or more training sessions, found by clicking on “Train your computer to better understand you”. Read aloud the text prompts to improve the Speech Recognition Engine’s performance at recognizing specific characteristics of your voice.

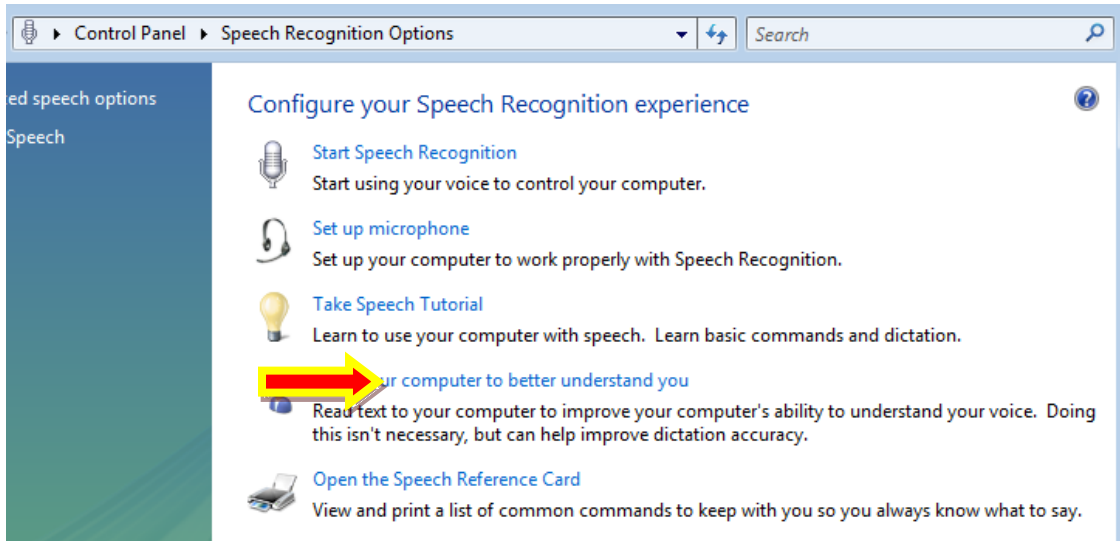


Figure 37 Speech Recognition Control Panel Options, initiating a training session

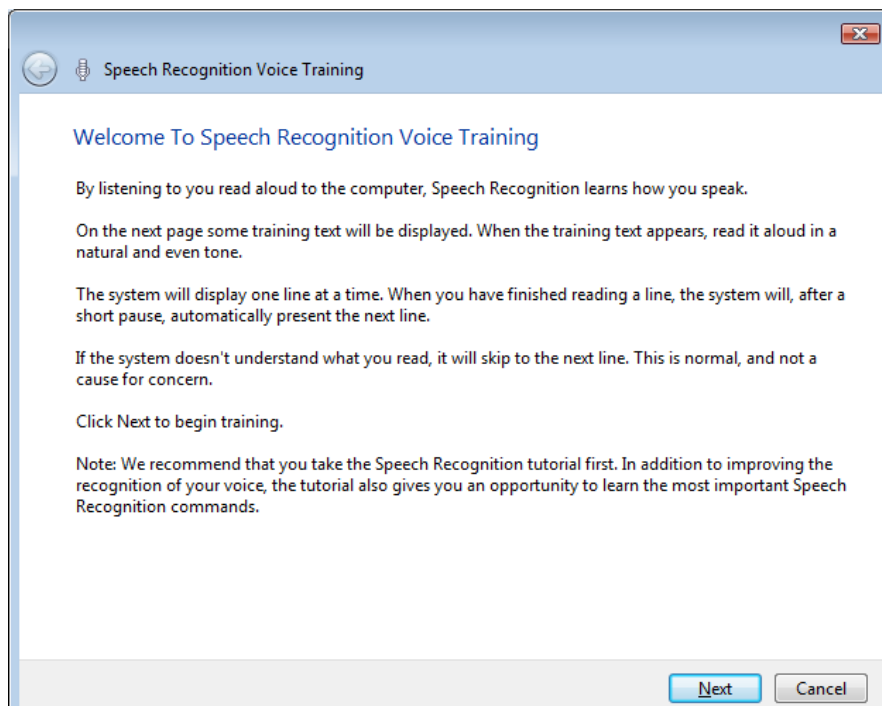


Figure 38 Speech Recognition Voice Training dialog

There are two training sessions that can be performed, and they may be repeated even after both are completed. The first one is called: "By listening to you read aloud to the computer, speech recognition learns how you speak." It begins with: "I am now speaking to my computer." And it ends with: "This concludes the tips for speech recognition training session". "To Read more about tips", "Look in the help documentation for speech".

The second training session begins with: "Speech is the main way for people to bond and learn from each other". And ends with: "The computer then enters this dictation into the program as text", and "This concludes the Speech Recognition background information training session". "To learn more about Speech Recognition", "Look in the help documentation for speech".

APPENDIX D: THE WINDOWS SPEECH DICTIONARY

Microsoft Windows Speech Recognition provides the means of interacting directly with the Speech Recognition Dictionary: pairs of words and their pronunciation, in order to enable successful rendering as text from spoken words. This technique is very helpful when words, such as “Theremin” do not seem to be present in the dictionary. The ability to add a word to the dictionary is a standard part of the Windows Speech Recognition feature (in Windows Vista/7).

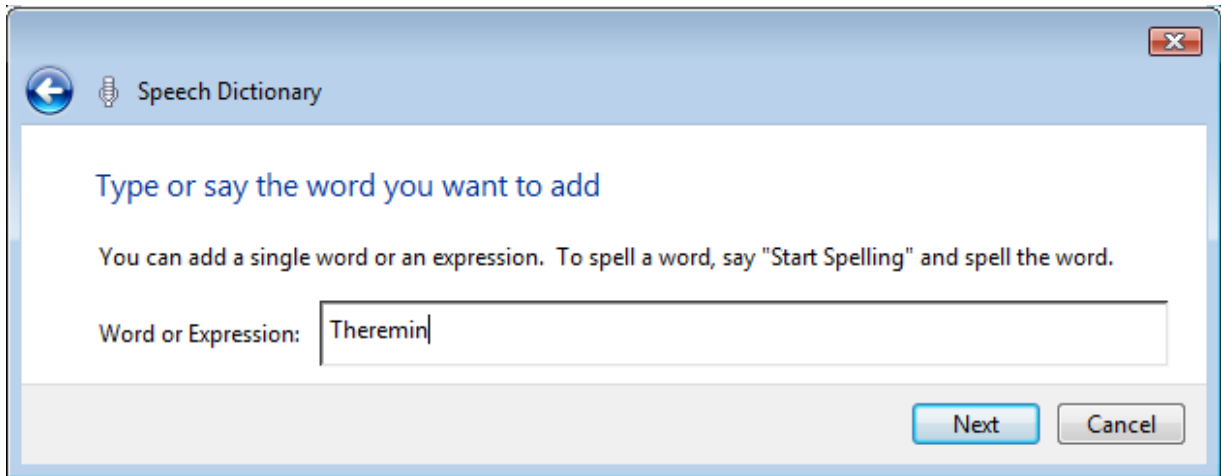


Figure 39 Adding a word to the Windows Speech Dictionary.

Near the end of the process, the user can provide a proper pronunciation of the word, by recording a spoken example.

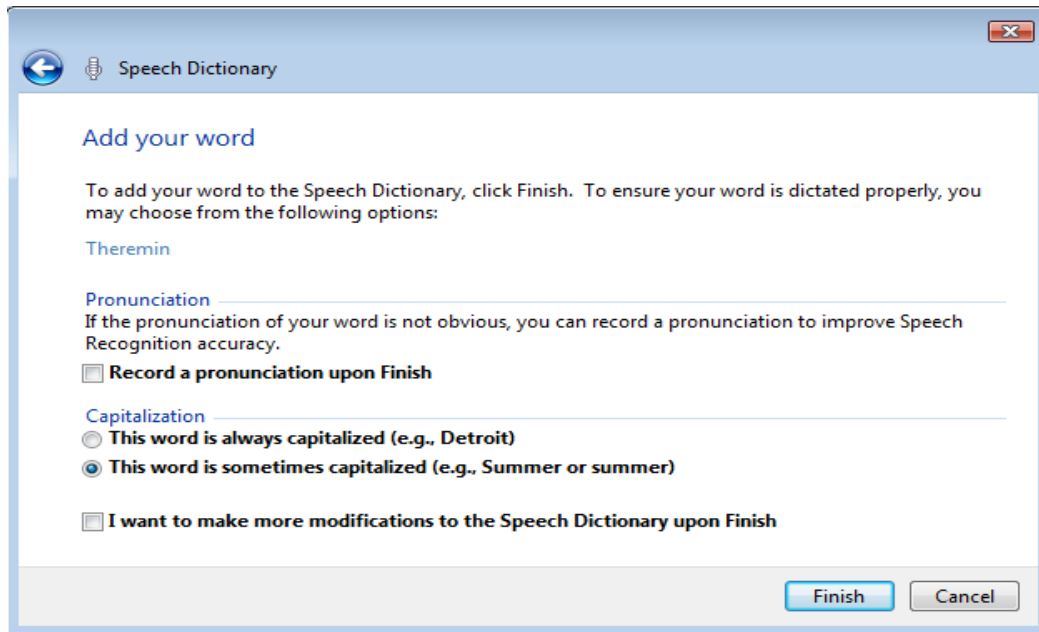


Figure 40 Record a pronunciation of new word added.

Ideally, this would be done automatically whenever a new name is not already in the dictionary, and has been deemed correct. However, we wouldn't want to add every mistaken name to the dictionary; only once the correct name is assigned. It is for this reason that the user must issue the command to get all the assigned names to load them into the actual grammar structure that the engine using. However, a way to automatically add words to the dictionary when needed was not discovered. Moreover, the ability to automatically prevent confused words from being recognized temporarily until the correct word is recognized and added to the list of choices in the loaded grammar, was also not discovered.

Prevent a Word from Being Recognized

When particular name is repeatedly misrecognized as a different word, for example, even after naming a track “Wow” and adding “Wow” to the Speech Dictionary, it gets repeatedly mistaken for the word “While”. The user can manually request that “While” not be recognized, so that “Wow” is correctly recognized. Figures 30 and 31 show how to prevent a word being dictated. Ideally this would be done (temporarily) when the user says “Wrong” after repeating a command using the same name, and getting the same wrong name.

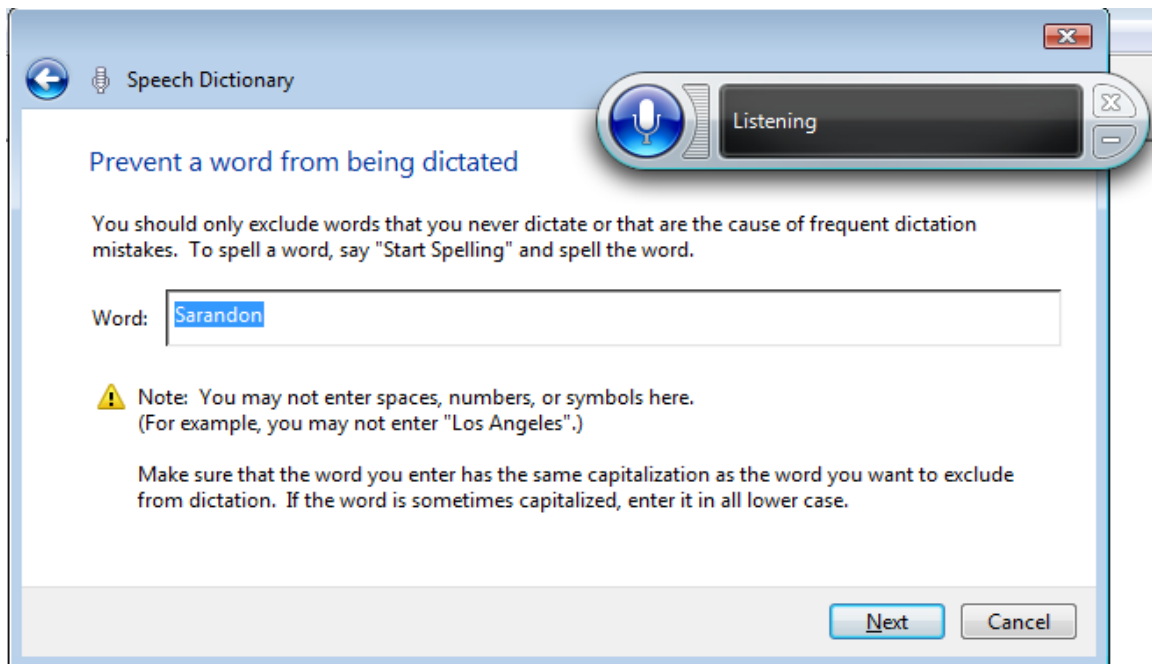


Figure 41 Prevent a Name from Being Recognized

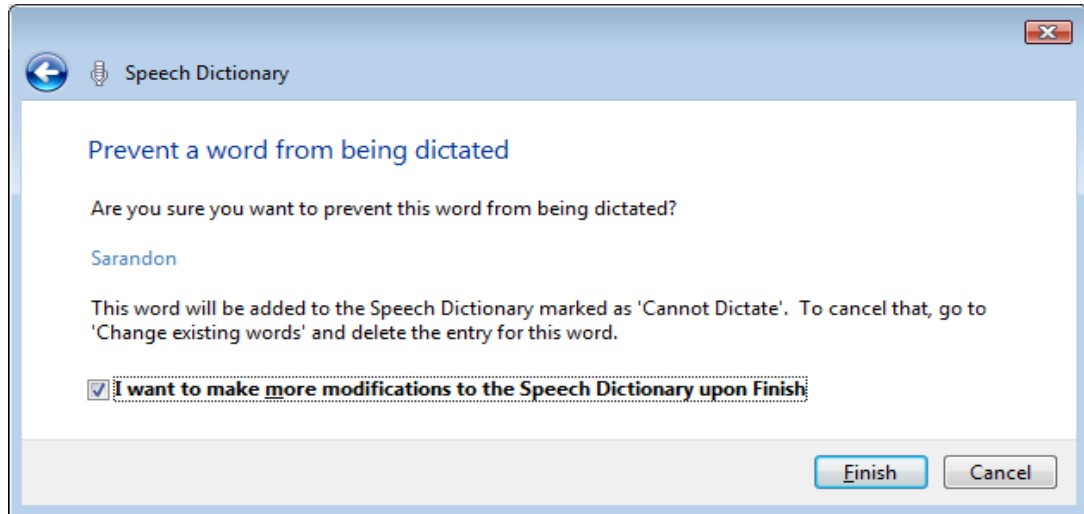


Figure 42 Prevent a word from being dictated.

Desired Function of ideal system: In the scenario of naming a track using a word not in the dictionary, the user must spell out the desired word, then once assigned correctly, save the session, and before adding the word to the list of possible track names, the word is tested for presence in the Windows Speech Dictionary for this user. If not present, then it is automatically added, and the user is given the chance to speak the word once, for the purposes of recording the correct pronunciation. In the scenario of preventing the recognition of a confused name: when a wrong name is assigned, the user says "Incorrect, the right name should be X", then the mistaken name is temporarily prevented from being recognized as dictation. So now, when the user repeats the command, a different, and hopefully the correct name will be recognized.

APPENDIX E: TEST PROCEDURE

Purpose: To measure the accuracy (as ratio of success to the total of success and failure) of the system at performing speech recognition commands. Various commands are exercised.

System Requirements: Windows Vista Business or Windows 7. Speech Recognition SAPI 3.5 or later, Visual Studio 2008, and the 2 projects: Audacity.sln and SayPlay.sln. Alternately, the experiment may be performed without Visual Studio by installing released versions of SayPlay and the custom modified version of Audacity.

Software Under Test: “SayPlay” is the program which accepts voice commands and “Audacity” is the audio recorder and editing software used to record and play tracks of audio.

Setup: In preparation for using Microsoft Windows Speech Recognition, please follow the instructions in Appendix C: Setting up for Speech Recognition. Perform one training session at the beginning of each testing session. Once the training session is complete, the speech recognition control panel may be closed. Remember, however, to return the selected speech recognition profile to a default

setting before any further use, in order to prevent the test subject's speech recognition profile from being corrupted by another speaker.

Launch Audacity and SayPlay, and verify the correct functioning of the voice command system by uttering a test "Play" command.

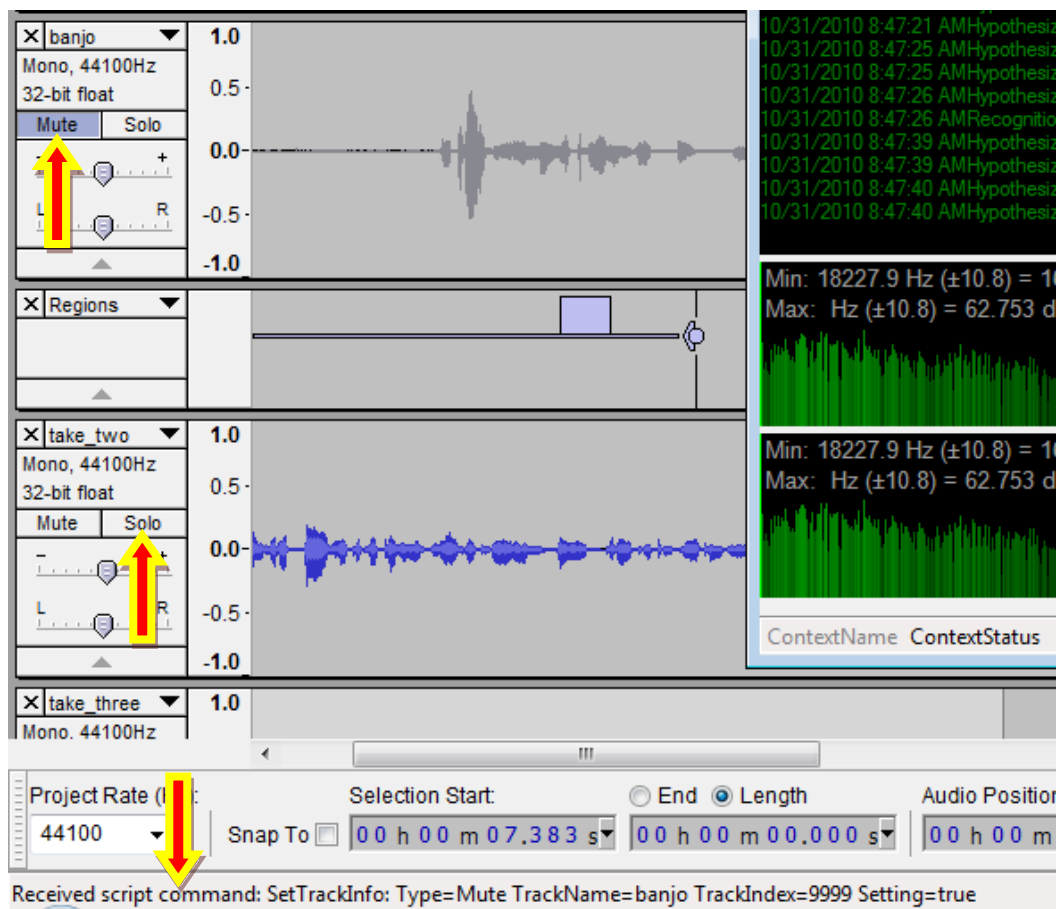


Figure 43 Audacity with custom SayPlay script status display

Each command can be verified received by Audacity when it is printed in the bottom status bar, as shown in Figure 43 Audacity with custom SayPlay script

status display: “Received Script command: Set TrackInfo:...”. For most commands the action of the command can be observed taking place.

Troubleshooting: If the system appears to be performing erratically, as in commands that had been readily recognized are now problematic, then it is likely that the microphone level has a setting which is too low, or too high. See Figure 32 Speech Recognition microphone setup for how to setup a microphone level.

After Conducting a Test: Remember, when finished testing, to return the selected speech recognition profile to a default setting before any further use, in order to prevent the test subject’s speech recognition profile from being corrupted by another speaker.



Figure 44 Windows Speech Recognition Profiles.

Test 1:	Basic transport commands	Time to Run Test: 12 mins
	Table 3 Test Plan	Baseline performance test, mean WSR Confidence
<p>Say “Play” to start playback.</p> <p>Wait a few moments and if it starts playback, say “Correct”. If after a few moments, nothing happens, say “Wrong, nothing happened”.</p> <p>Once the system is playing back, say “Stop”. If playback stops, say “Correct”. If after a few moments playback does not stop, say “Wrong”.</p> <p>Repeat > 25 times, for Play, Stop, Record and Save also.</p> <p>Notes:</p> <p>When saying “Wrong” to indicate a failed command, one may also add a comment about what went wrong, such as “nothing happened”, or “The two previous commands were not followed”.</p> <p>Sometimes a user will forget to say “Correct” when a command is usually right. In this case the subsequent following command is given.</p> <p>Sometimes saying “Wrong” is forgotten when a command is sometimes wrong. In this case the command is repeated.</p> <p>Sometimes the judgment of Correct or Wrong is rendered more than one command later, because, as mentioned in the examples above, it was correct, so it was not repeated, or it was incorrect, and so it had to be repeated.</p>		

Table 8 Test 1 Procedure: Baseline Measure

The next test exercises a much larger set of commands available through the Audacity scripting interface. Each command should be exercised at least 10 times. The resulting log file is sorted by command name, and the confidence values are averaged per command, and sorted from highest to lowest. The goal is to determine the best choice for the confidence threshold. This is the value above which the confidence value returned with the recognition event from the speech recognition engine must be, in order for the command to be acted upon. Note that this is not the same as the statistical measure called the confidence

interval. It is part of the recognition event data structure returned from the speech recognition engine.

These command names could be changed in software, to something easier to remember, but each user has different preferences, and so the ideal solution is to allow the user to state their desired name for each command. This feature was not implemented and is described in the “Future Work” section of Chapter 5.

Test 2	Table 3 Test Plan	Figure 10 Mean Confidence value returned from WSR Engine
Issue each command multiple times, recording pass/fail status		
Command	Audacity Script	Desired Effect
play	Play	Begin playback
stop	Stop	Stop recording/playback
record	Record	Begin recording
pause	Pause	Pause playback
save	Save	Save session document
undo	Undo	Undo last command
redo	Redo	Redo last command
new	New	Create a new session
cut	Cut	Remove selected audio
split cut	SplitCut	
copy	Copy	
paste	Paste	Insert audio from scrap
trim	Trim	
delete	Delete	
split delete	SplitDelete	
silence	Silence	Selected audio should silence
split new	SplitNew	
join	Join	
disjoin	Disjoin	
duplicate	Duplicate	
skip start	SkipStart	Play cursor returns to start
skip end	SkipEnd	Play cursor jumps to end
previous track	PrevTrack	Focused track moves up one
next track	NextTrack	Focused track moves down one
mute all tracks	MuteAllTracks	
unmute all tracks	UnMuteAllTracks	
unsolo all tracks	UnSoloAllTracks	
cursor short jump left	CursorShortJumpLeft	cursor jump left ~5 seconds
cursor short jump right	CursorShortJumpRight	cursor jump right ~5
cursor long jump left	CursorLongJumpLeft	cursor jump left ~15 sec
cursor long jump right	CursorLongJumpRight	cursor jumps right ~15
set left selection	SetLeftSelection	
set right selection	SetRightSelection	
select start cursor	SelStartCursor	
select cursor end	SelCursorEnd	
zero crossing	ZeroCross	cursor to nearest zero crossing
cursor select start	CursSelStart	

cursor select end	CursSelEnd	
cursor track start	CursTrackStart	
select extreme left	SelExtLeft	
select extreme right	SelExtRight	
select set extreme left	SelSetExtLeft	
select set extreme right	SelSetExtRight	
select center left	SelCntrLeft	
select center right	SelCntrRight	
track gain increment	TrackGainInc	Focused track gain increments
track gain decrement	TrackGainDec	Focused track gain decrements
silence labels	SilenceLabels	
split labels	SplitLabels	
disjoin labels	DisjoinLabels	
select all	SelectAll	
select none	SelectNone	

Table 9 Test 2 All Keyword Commands

Test 3: Track Commands	(Un)Mute and (Un)Solo by name. Tests run on many different test subjects, w/and w/out accents.	Time to Run Test: 12 mins
Say (Un)Solo/(Un)Mute the X track, where X = { piano, guitar, bass, vocal, drums, snare drum, hi hat, kick drum, crash, ride, percussion, trumpet, clarinet, flute, horn, trombone, saxophone, alto sax, tenor sax, soprano sax, accordian, trumpet, harpsichord }		
Also Say “UnMute All Tracks” and Say “UnSolo All Tracks”. NOTE: The command “Unsolo All Tracks” does not work while in playback.		

Table 10 Test 3 Test Procedure: Field Trials

Test 4: Adding Tricky Names to the Windows Speech Dictionary
Try to issue commands on the following tracks by name: { Crotales, Gambales, Wow, While } Then, add the name to the Windows Speech Dictionary Repeat the action of issuing track names as above. Record Success/failure verbally using “Pass”, or “Correct”, and “Fail”, or “Wrong”

Table 11 Test 4 Test Procedure: Tricky Names

Test 5: Add Theremin to Dictionary & Grammar	Actions: (Navigation Table 3 Test Plan)
Name a track "Theremin"	
Assign with elaboration	"Name this track Theremin like the Russian inventor Science Fiction Movie Sound, Good Vibrations.
Refer to name	(Un)Mute, (Un)Solo, Pan, Theremin
Prevent Dictation of confused name "Sarandon"	See Appendix for how to prevent dictation of words.
Add Names to Grammar	"Computer, Please Refresh the Session"
Commands which 5Refer to name accuracy	(Un)Mute, (Un)Solo, Pan, etc.
<p>Name a track "Theremin" then Perform several track commands (Mute etc.) on Theremin track</p> <p>Prevent dictation of the frequently confused word "Sarandon" and repeat the performance of several track commands on the Theremin track (Mute the Theremin, etc.)</p> <p>Add the word "Theremin" to the Windows Speech Dictionary, and measure the results as above.</p> <p>Add the track names into the running grammar, and repeat commands as above.</p>	

Table 12 Test 5: Effectiveness Adding Theremin to Dictionary and Grammar

Test 6: Adding to Dictionary & Loading Grammar	Action (Table 3 Test Plan)
Name a track from the list of tricky names	"Name this track X"
Refer to name	(Un)Mute, (Un)Solo, Pan, Theremin
If (obscure) name is never recognized, add it to Dictionary	
Add Names to Grammar	"Computer, Please Refresh the Session"
Commands which Refer to Tracks by name:	(Un)Mute, (Un)Solo, Pan, etc.
<p>Name a track the known tricky name from the following list: Alto sax, Bass, Cowbell, Crotales, Djembe, Kazoo, Raisins, Sticklavier, Tapping, Theremin, Violins, Vocal scat, Wow</p> <p>Perform ~3-5 track commands on the each track (Mute the Theremin, etc.)</p> <p>If there are no positive recognition events, add the word to the Windows Speech Dictionary.</p> <p>Add the track name into the running grammar, and repeat commands as above ~15 times.</p>	

Table 13 Test 6: Adding Tricky Names to Dictionary and Grammar

Test 7: Elaboration & Loading Grammar	Action (Navigation: Table 3 Test Plan)
Name a track from homophone list	Name this track X (or Y).
Result is deemed to be the default	
If failure try assigning with elaboration	Name this track X like Y
Refer to name with Track Commands	(Un)Mute, (Un)Solo, Pan, etc.
Name track from homophone list alternate	Name this track A
If fails, Assign with elaboration from list	Name this track A like B
If assignment fails, use Spelling it out	Name this track spelled a.b.c.d.etc.
Refer to name with Track Commands	(Un)Mute, (Un)Solo, Pan, etc.
Add Names to Grammar	Computer Please Refresh Session
Refer to name with Track Commands	(Un)Mute, (Un)Solo, Pan, etc.
<p>Purpose of Experiment: Use homophone pairs as input in assigning names, to test whether elaboration is effective. Homophones sound the same, but are spelled differently. One or the other should be recognized when speaking the sound for both. Elaboration may allow selection of one or the other. In addition, loading the desired spelling into the grammar may provide correct recognition of the desired spelling of the word.</p> <p>Background Information: In general, without adding any extra elaborating words, one spelling of the homophone pair will be recognized more than the spelling. This dominant result is called the “Default” result for the homophone. The alternative spelling of the word (the one not generally recognized) is called the “non-default” spelling for the purposes of this experiment.</p> <p>Elaboration (adding some extra words, using “Like” or “As In”) could cause the likelihood of recognition to tip one way or the other, and to select either the default or the non-default sense of the homophonous word, depending on the associated words used in elaboration.</p> <p>For example, saying “Name this track bass, as in bass guitar”, should select “Bass” over “Base”.</p> <p>The overall steps of the experiment involve determining which spelling is the default, and then trying to obtain accurate recognition of the non-default spelling of the word by using elaboration.</p> <p>If no positive recognition of the non-default spelling of the homophone pair can be found using elaboration, then the name is assigned by spelling it out.</p> <p>For example, using the homophone pair Presence/Presents, the user simply commands: “Name this track (presence/presents)” (Remember both sound the same). The result (“Presents”) is deemed to be the default. Next, the experimenter attempts to assign the non-default spelling of the word “Presence”, using elaboration: “Name this track presence like in attendance”. If this is successful,</p>	

then this is a case of successful elaboration, since the recognition result was steered away from the default and to the non-default result by elaboration.

If all attempts at elaboration fail to result in a correct recognition of the non-default spelling of the homophone pair under test, then the non-default form is assigned by spelling it out. For example, after all attempts at naming the track “Lyre” fail using elaboration (Name this track Lyre as in the ancient Greek harp) then the experimenter says “Name this track spelled L-Y-R-E”. Once this is done correctly, the non-default form of the homophone can be loaded into the grammar as a legitimate track name choice. The command to load all track names into the grammar is: “Computer, please refresh the session”.

Now that the non-default spelling of the homophone has been assigned and loaded into the grammar, so that the speech recognition engine doesn’t have to rely on dictation speech recognition, it is tested for accuracy.

Several commands are then issued on the track by name, and successful results are tallied. This demonstrates whether or not the non-default spelling of the homophone pair can be used.

When issuing the command, the dominant word sense recognized is called the “default”. For example, if we use the Base/Bass homophone pair, and “Base” is commonly recognized, then Base is considered the default spelling of the homophone pair Base/Bass.

Elaboration is then used to attempt successful recognition of the *non-default* spelling. If all attempts fail, then we force the name assignment by spelling it out.

Once the non-default name has been assigned by spelling it out, commands are attempted on it. The expectation is that the speech recognizer continues to recognize the homophonic word as the default.

This is when we add the non-default name to the running grammar. Several commands issued on the track now should be successful. For statistical significance, there should be at least 10 successes in a row. This illustrates the improving effect of having the words be loaded into the grammar, and in fact, it doesn’t work at all otherwise.

Note that it is not possible to load both homophonic words into the grammar, because they sound exactly the same, and one would get all the recognition hits.

Steps to Reproduce Experiment:

For each pair of homophones in the associated list below, do the following:

Name a track from the list of homophones

Identify which spelling is the “Default” recognized spelling. This is the resulting name which occurs most often. If it is neither of the two homophones, then indicate Default is “neither”.

Initiate several (~10-20) commands on the “default” named track, to see whether dictation speech recognition continues to identify the “default” spelling.

Attempt to rename the track the non-default spelling by using the suggested elaboration listed for that spelling of the homophone pair.

If this fails after several attempts, and try any other elaboration words that come to mind, keeping in mind the elaboration phrasing with “Like” and “as In”

If all attempts at using elaboration fail to assign the track to the non-default name, then assign the non-default spelling of the homophone by spelling it out.

Now, we will add the non-default name to the loaded grammar, to determine whether what never worked before will now work effectively.

Perform at least 10 trials to gather enough data for statistical significance.

Intended	Homophone	Elaboration	Notes
Bass	Base	Like bass guitar	Base is default, elaboration has worked
Oracle	Auricle	Like a seer	Oracle is default, elaboration for Auricle didn't work
Airy	Aerie	Like floaty, fluffy	Not a good test case
Aural	Oral	Like halo of energy	Not a good test case
Awful	Offal	Like awfully bad	Not a good test case
Away	Aweigh	Anchors aweigh Going away	Sometimes aweigh appeared in elaboration, but not for name
Band	Banned	Group / Banished	“and” was most popular result
Bass	Base	Like bass guitar, or first base	
Basses	Basis	Like bass guitars	Had to spell it to get it to stick
Bated	Baited	Like bated breath Like baited trap	Name as follows: bated worked
Beat(s)	Beet(s)	Like drum beats Like pickled beets	“eat” was a wrong result. Beats (plural) also detected.
Bell(s)	Belle(s)	Like silver bells Like southern belles	Belle worked wonderfully once it was loaded into the grammar (once spelled)
Bowed	Bode	Bowed strings Bode well	Bode was recognized. Once spelled /loaded in grammar, “bowed” worked
Cessions	Sessions	Territorial cessions/ recording or therapy	
Chords	Cords	Like piano chords	Cord singular, chords plural (default)

		Like mic cords	
Colonel	Kernel	Colonel Sanders Popcorn kernel	Colonels frequently
Chanty	Shanty	Like a sea chantey shanty town	Shanti (even though it spell check won't accept) was default.
Choir	Quire	Preaching to / quire of paper	Check logfile
Chorales	Corrals	Bach chorale / horse yard	Bach chorale couldn't cause correct, and corral was default.
Choral	Coral	Choral arrangement Coral reef	Check logfile
Cymbals	Symbols	Crash, ride, hihat, splash, Chinese, Runes	Never got cymbals to stick: try spelling it: worked every time.
Forte	Fort	Like fortissimo or for'tay	Worked
Four	Fore, for	Number four/foreground	... fore track worked (thought it'd be "four track")
Hey	Hay	Hey there, hay is for horses, haystack, hay bales,	Didn't load grammar with "Hey".
Halves	Haves	Half dollars/have-nots	Similar results to others, halves by default, loaded haves after spelling it and it worked well.
Hertz	Hurts	Kilohertz/ pains	Hurts like pain elaboration worked for naming track
Hymn	Him	Church hymn/his person	"Him" worked a tiny bit w/o loading. Hymn never did until loading (had to spell it out)
Lyre	Liar	Greek harp/untrustworthy	Had to spell out lyre, but once added to grammar it worked every time. Liar (default) worked from start
Lute(s)	Loot(s)	Renaissance guitar/booty	
Minor	Miner	Youth / gold digger	Minor worked somewhat without loading grammar Miner as in a forty niner worked as elaboration
Pairs	Pears	Groups of two / fruits	Peres was the mystery word it kept coming up with
Paired	Pared	Groups of two / trimmed	Check logfile
Pause	Paws	Rest / dogs feet	Check logfile
Peals	Peels	From a bell / From an Apple	Peels from an apple

Presence	Presents	In attendance / gifts	Elaboration for “like in attendance” works, whereas prior it defaulted to presents.
Rap	Wrap	Rap sheet, rap music/ gift wrap	Wrap by default, “As in rap sheet didn’t work at first, but later it did, rap music did work.
Rest	Wrest	Pause / take away	Rest worked some by default (no loading) Wrest worked well after loading into grammar
Rhyme	Rime	Poetry / ancient mariner (frosty crust)	Neither word worked by default: orion, ryan, Elaboration didn’t work either
Rite	Right	Rite of spring, ritual / Legal	Right worked by default. Rite worked after loading into grammar. No elaboration worked.
Root	Route	Source / Road	Route by default. And worked with solo and mute but not with pan, didn’t load. Loaded Root, and worked very well.
Sink	Sync	Kitchen basin / lip sync	Sync with elaboration produced saint instead worked well after loading grammar
Staff	Staph	Musical staff / infection	Staph did not get loaded into the grammar, even after trying twice. Put the system to rest to return later.
Stile	Style	Turnstile / Fashion	Style is default
Timbre(s)	Timber(s)	Sonority / Felled trees	Timber default, timbre is pronounced “tamber”, and works some after spelling w/o loading grammar
Tic(s)	Tick(s)	Tic toc, tick tick	Tick is default,
Time	Thyme	Time signature, French thyme	Time is default.
Tooter(s)	Tutor(s)	Horn / teacher	Neither were recognized, no default
Undo	Undue	Menu command / undeserved	
Way	Weigh	As in to measure the weight	Wei was an answer, a proper name
Weakly	Weekly	As in feeble	
Wood	Would	As in wooden	Good Elaboration: “as in wooden”
Note: Highlighted text indicates the “default” spelling of the recognition result.			

Table 14 Test 7: Homophone Tests of Elaboration and Loading into Grammar.

Notes on annotations of the logfile data: Failure categories are now augmented to include “Other Sense of the Word”, where the intention was the other spelling. Without elaboration, this is unfair to assign this designation, since without elaboration there is no way to know the user’s intentional spelling. This failure designation is assigned only when elaboration fails and the recognized result is the corresponding homophone pair.

APPENDIX F: SUMMARY OF RULES APPLIED TO WILDCARD
TEXT STRINGS

Whenever dictation speech recognition is used to fill a slot with wildcard text converted from speech, such as in the slot for a name being assigned, a string of text is returned from the Windows Speech Recognition engine. This string of text may have problems, for which the following rules have been useful in overcoming. These rules are applied only to the string of wildcard text recognized to be the name being assigned to an audio track.

The following table shows each rule, the problem it seeks to solve, and the effectiveness in solving that particular problem, and any side effects.

Rule	Example	Problem Solved
“Like”, “As In”, “Is In”, “Is An”, “As An”.	“Name this track Theremin like the Russian Inventor” or “Name this track ‘bass’ like bass guitar”	Allows names to include any of the words normally stricken by the application of the other rules
Detect “As Follows” and keep all text following to use as the name.	“Name this track as follows: ‘Scream like a banshee’ ”	Allows names to include any of the words normally stricken by the application of the other rules
Leading “The” is removed from a name string.	If a track is named: Name this track “the piano track”, then Mute the “piano” track is recognized, not mute “The Piano” track. So the name is not recognized with a leading “The”.	“The” cannot be part of a name, because it is recognized as an optional part of the grammar. Such a name will never be recognized because “the” will always be assumed to be part of the grammar, not the name
Trailing words: “I”, “Oh”, “Ah”, “And”	“Name this track lead vocal I”	The “Breath After” problem of appending a nonsensical word misrecognized from a trailing dysfluency, like taking a breath

Table 15: List of Rules Applied to Names Strings When Assigning Them.

APPENDIX G: SYNOPSIS OF EXPERIMENTS

Experiments exercise voice command functions and the user notes whether a correct response results. Success rates and Confidence values are averaged, and the causes for failures are attributed.

Experiment	Purpose	Results and Interpretation
Baseline Performance	Establish a Baseline of performance for basic commands	Shows the importance of proper microphone setup and signal level
Keyword Commands	Establish a Baseline of performance for all Keyword commands	Shows that the Speech Recognition Confidence level can be high enough
Field Tests with Multiple Users	Field testing of prototype in simulated recording sessions	Shows that eventually even for speakers with heavy accent, it works. Shows that Keyword Commands are better recognized than Name Commands, and Wrong Name happens a lot.
Add Tricky Names to Dictionary	Exercise known tricky names	Shows that adding tricky names to the Speech Dictionary helps, especially for tricky names
Successive Improvements:	To demonstrate increases in recognition accuracy performance when using available methods.	Shows that performance improves when adding a name to the dictionary, again when preventing confused words and again when adding names to the grammar.
Additional Tests on 20 tricky names	To Isolate each feature and measure performance increase	
Elaboration Spelling it out Loading into Grammar	Try to find a pattern in what works Measure recognition accuracy Without creating names	Elaboration can help to assign names When names are loaded into grammar, much better recognition accuracy results.

Table 16: Overview of Experiments and Results

Baseline Performance

Purpose:

Gain an understanding of performance in order to set the threshold of confidence required to take action (to issue a command).

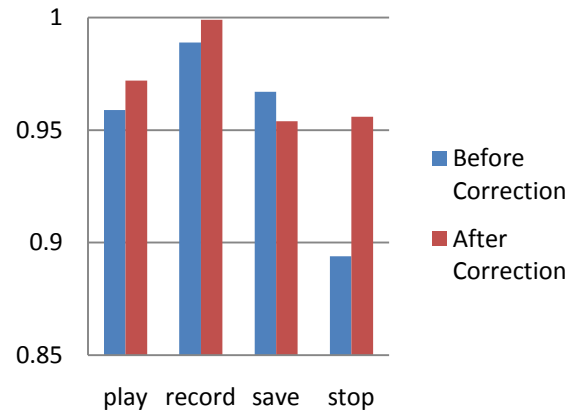
Interpretation:

The four basic keyword commands: Play, Record, Stop and Save each uttered >25 times, and the average of the Confidence Score from the Speech Recognition Engine is plotted. The microphone level was set to a low value initially (Shown in Blue), and was corrected, afterward, (Shown in Red).

Conclusion:

The Confidence Threshold should be set to around 0.93.

Confidence before/after mic setup



Keyword Commands

Purpose:

To show that command recognition is reliable: each keyword command was issued >25 times and the average Confidence value returned from the Speech Recognition Engine is plotted for each command.

Interpretation:

None of the commands averaged lower than the chosen threshold of 0.93, for issuing commands.

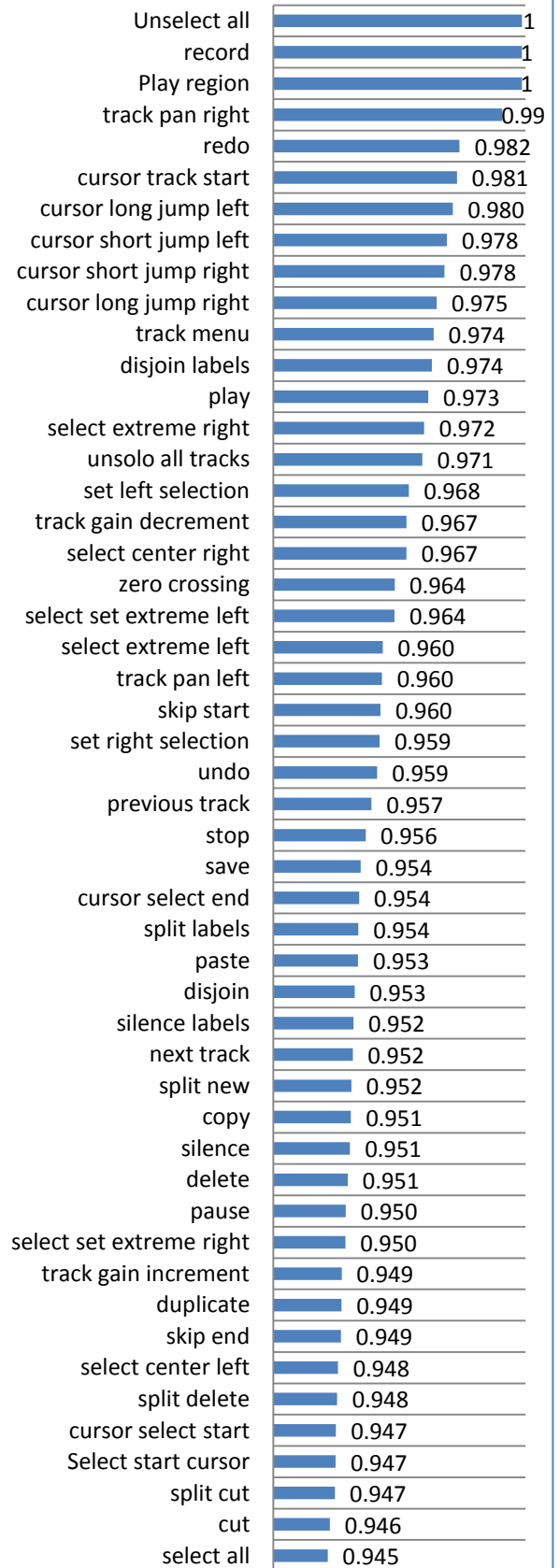
Some commands were issued more times than others, and they tended to average higher scores.

Conclusion:

The high averages of Confidence show that the threshold can be set fairly high and reliable speech recognition commands will result.

The higher scores for more utterances is most likely due to the Machine Learning that takes place, where the more a command is issued, the higher the confidence that it is recognized correctly when it is recognized.

Command



Field Tests with Multiple Users

Purpose:

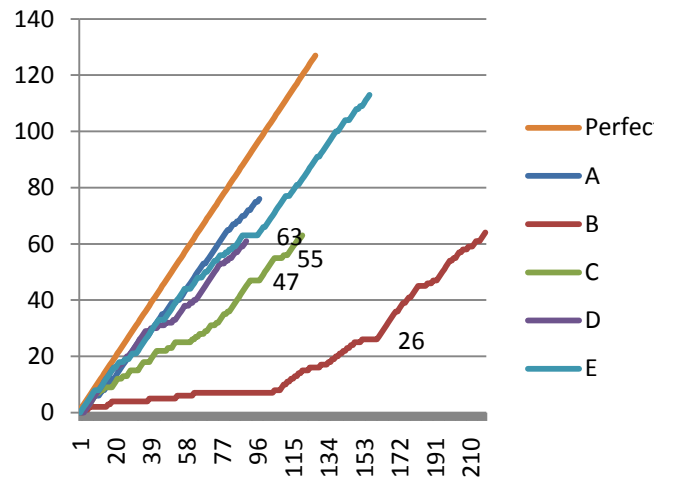
Plot cumulative successes to show improvement in recognition accuracy over time.

Interpretation:

While some sticking points were found, such as on certain words like “Vocal Scat”, or Alto Sax, the speech recognition accuracy can improve.

Conclusion:

Speech recognition accuracy can improve with use, to become reasonably reliable.



Purpose:

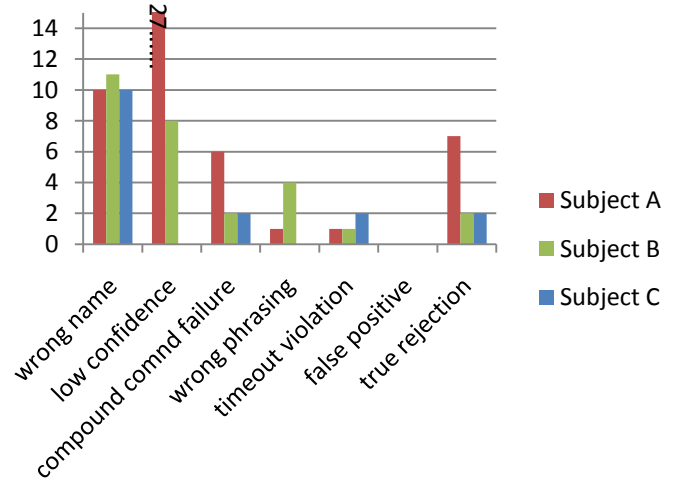
From the above test data, find the likely failure modes.

Interpretation:

After Low Confidence, the Wrong Name is the most prolific failure mode. Examination of the data shows the majority of low confidence scores as coming from a test subject who speaks with an accent.

Conclusion:

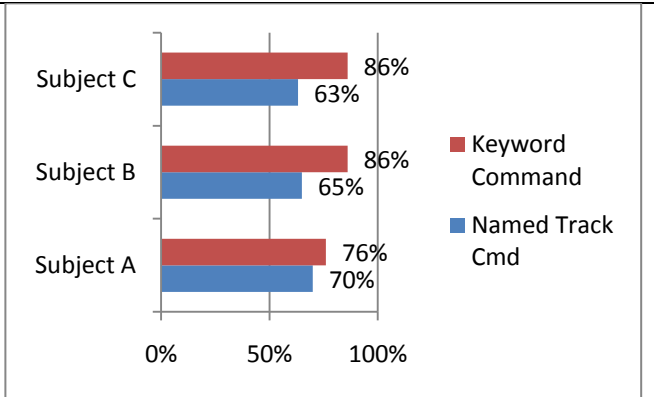
Attacking and improving the relatively low accuracy of recognizing names will make the system better.



Purpose:
 From the above test data, categorize the recognition accuracy rates as either Keyword Commands or as Name Commands.

Interpretation:
 The recognition accuracy of Named Track Commands is consistently lower than for Keyword commands, for the 3 best performing test subjects.

Conclusion:
 Improving accuracy of Named Track Commands will result in a better performing system.



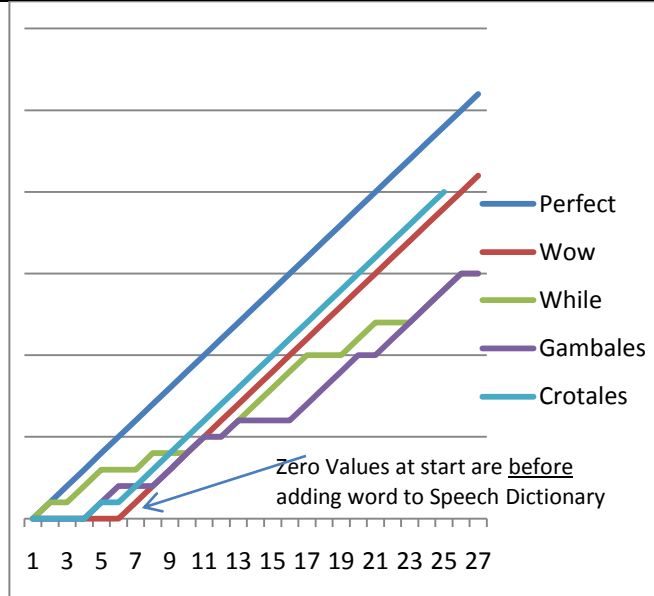
Add Tricky Names to Dictionary

Purpose:
 Test the effect on recognition accuracy of Adding Tricky Names to the Speech Dictionary.

Add to Speech Dictionary, Prevent Recognition of Confused Words, Add to Loaded Grammar

Interpretation:
 Issuing track commands on Wow, Gambales and Crotales (words not in the Speech Dictionary) show no successes until after adding these words to the Speech Dictionary. Then, there were significantly higher success rates for each of them.

Conclusion:
 If a word is not in the Speech Dictionary, then it will not work at all. If it gets added, then improvement ranges from working much better to working flawlessly.



Successive Improvements:

Purpose:

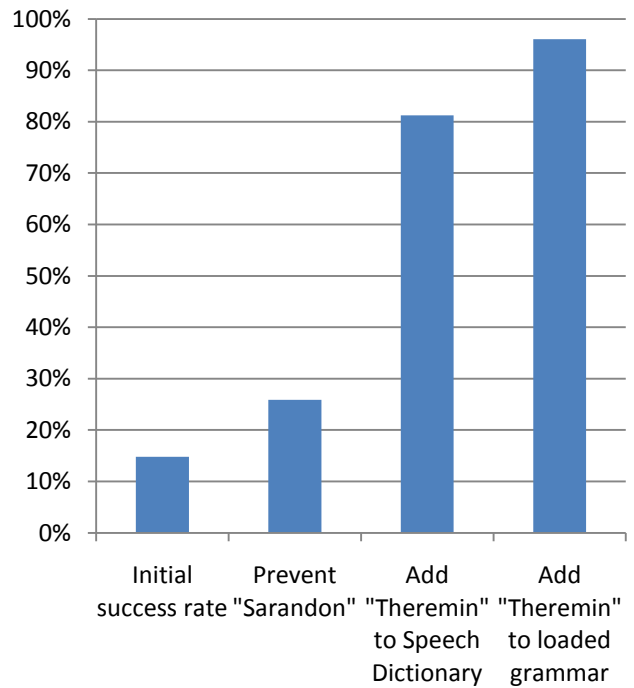
Determine the relative influence of preventing recognition of confused names, adding names to the Speech Dictionary, and loading them into the Grammar, on recognition accuracy.

Interpretation:

Why was the initial success rate non-zero if the word was not in the Dictionary? It was in the Dictionary, but adding it provides a chance to pronounce it. This act is what causes the jump in improvement after adding "Theremin" to the Dictionary.

Conclusion:

These techniques help speech recognition accuracy.



Load Theremin into Grammar

Purpose:

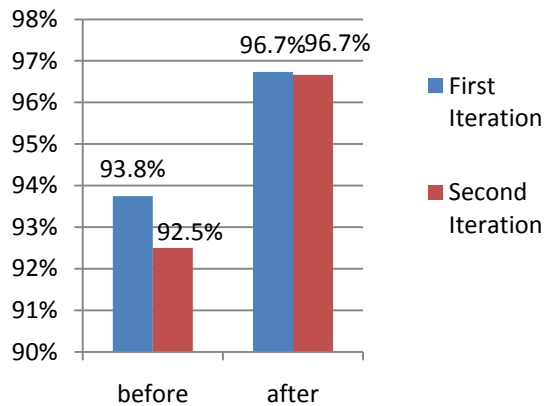
Isolate the improvement in recognition that is due to loading names into the grammar. Also determine whether adding the word to the Speech Dictionary helps.

Interpretation:

For every word tested, adding the word to loaded grammar improved recognition accuracy.

Conclusion:

Loading "Theremin" into the grammar provides measurable improvement in recognition accuracy.



Elaboration using Homophones

Purpose:

Use Homophones to show whether Elaboration works to recognize a more obscure spelling of a homophone word. For example, recognizing “Lyre” over “Liar”.

The set of homophones as input was drawn from a list of about 2000 homophones, filtered down to those which may have some connotation with recording.

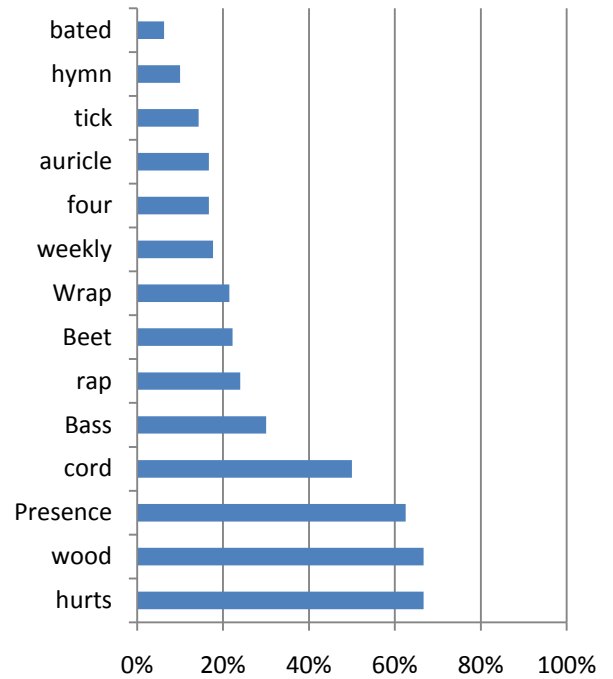
Interpretation:

Elaboration does not often work to assign a name to a track. Some interesting specifics were learned, about the tendency to favor proper names over ordinary words, such as “Wei” over “way”. The few times where it clearly worked were

Conclusion:

Elaboration requires further refinement in order to be more useful. The user must think differently in order to use it more effectively, by coming up with extra words that are frequently heard together with the desired word, rather than simply redefining the word.

elaboration success



Homophones to Show Effectiveness of Loading Names into Grammar

Purpose:

Use Homophones to show whether Elaboration works to recognize a more obscure spelling of a homophone word. For example, recognizing “Lyre” over “Liar”.

Since homophones can be recognized multiple ways, and the non-default way is loaded, it is the

The set of homophones as input was drawn from a list of about 2000 homophones, filtered down to those which may have some connotation with recording.

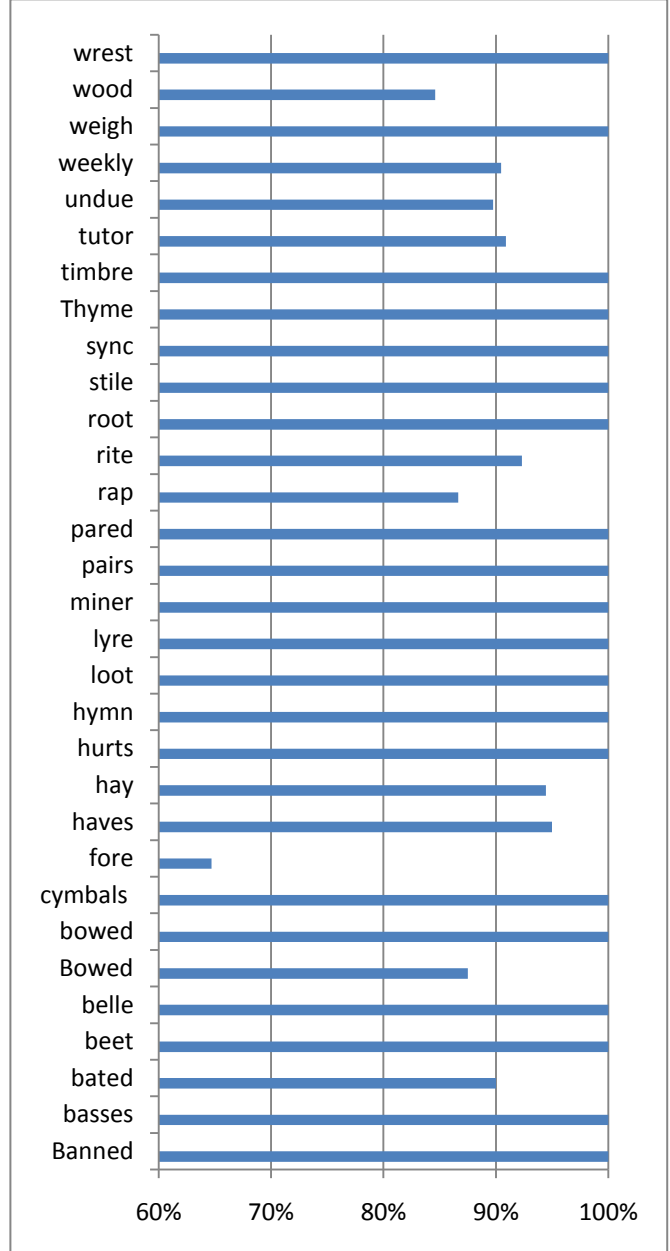
Interpretation:

While Elaboration does not often work to assign a name to a track, spelling it out almost always works. And, loading the name with the obscure spelling into the grammar allows even the more obscure spelling of the homophone pair to be accurately recognized.

The poor results for “Fore” may have been attributed to 2 other homophones, For and Four. The latter word being a number, is already part of the grammar for numbered tracks, which explains several mis-recognition events favoring “Four” (or fourth).

Conclusion:

Loading words which are part of homophone pairs into the grammar provides a high level of recognition accuracy for most homophones tested. The experiment focused on loading the more obscure name into the grammar, to ensure it would be recognized over the more prominent (default) spelling of the word.



APPENDIX H: BIBLIOGRAPHY

"The market for speech applications in mobile computing expected to triple by 2014", 2009, Database and Network Journal, vol. 39, no. 3, pp. 19.

Annotations: Mobile Handset applications increase from \$32.7 million to \$99.7 million. Navigation (GPS) devices. Network based ASR (where speech is processed remotely from the raw PCM audio) versus embedded ASR (where speech dictionary and algorithm processing power and memory are limited to the mobile device). Applications for inventory control are described.

Ananthakrishnan, S. & Narayanan, S.S. 2008, "Automatic prosodic event detection using acoustic, lexical, and syntactic evidence", IEEE Transactions on Audio, Speech and Language Processing, vol. 16, no. 1, pp. 216-228.

Annotations: Detection of speech events among non-speech events is critical for reducing false positive events. The application of this research may help to address the "timeout error" problem. If the more effective distinction between speech and non-speech sounds works, this may allow an increase in the timeout parameter so that a speaker can pause longer when in the middle of a recognition phrase.

Ayres, T. & Nolan, B. 2006, "Voice activated command and control with speech recognition over WiFi", Science of Computer Programming, vol. 59, no. 1-2, pp. 109-126.

Annotations: Uses SPHINX II.

Baker, J., Deng, L., Glass, J., Khudanpur, S., Chin-hui Lee, Morgan, N. & O'Shaughnessy, D. May 2009, "Developments and directions in speech recognition and understanding, Part 1", IEEE Signal Processing Magazine, vol. 26, no. 3, pp. 75-80.

Baker, J., Deng, L., Glass, J., Khudanpur, S., Chin-hui Lee, Morgan, N. & O'Shaughnessy, D. July 2009, "Updated MINDS Report on Speech Recognition and Understanding, Part 2", IEEE Signal Processing Magazine, vol. 26, no. 4, pp. 78-85.

Annotations: Describes areas of progress and challenges in Automatic Speech Recognition (ASR) in its many form factors: Dictation, command & control, dialogue systems, and conversational systems.

Caskey, S. 2009, "Spoken language input for a patient note system", HEALTHINF 2009. Second International Conference on Health Informatics, pp. 323.

Chang, J. 2009, "Usability evaluation of a Volkswagen Group in-vehicle speech system", Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications, Automotive UI 2009, pp. 137.

Annotations: Test methodology and results are well described.

Chun-Liang Hsu 2009, "Constructing intelligent living-space controlling system with blue-tooth and speech-recognition microprocessor", Expert Systems with Applications, vol. 36, no. 5, pp. 9308-18.

Chun-Liang, H. 2008, "Implementing speech-recognition microprocessor into intelligent control-system of home-appliance", 2008 IEEE Asia-Pacific Services Computing Conference (APSCC 2008), p. 881.

Annotations: Discusses modes of operation for controlling multiple home electronic devices, such as the telephone, internet or appliances. Speech Recognition comes on an ASIC, and is implemented into a hardware system.

Cohen, J. 2008, "Embedded speech recognition applications in mobile phones: Status, trends, and challenges", 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, March 31, 2008 - April

04Institute of Electrical and Electronics Engineers Inc, Las Vegas, NV, United states, pp. 5352.

Annotations: Mobile Phone history, Word Spotting Speech Recognition, recommends that a “killer application” be developed to promulgate the technology.

Cook, Brad; 4 February 2004, MacSpeech releases iLife '04 Voice Solution for iListen; MacCentral: accessed 3 May 2011
<http://www.macworld.com/article/29423/2004/02/macspeech.html>

Annotations: A ScriptPak provides keyword command recognition for menu commands. Each are application specific. One was released for GarageBand, the music recording application.

De Mori, R. 2008, "Spoken language understanding", IEEE Signal Processing Magazine, vol. 25, no. 3, pp. 50-8.

Annotations: Good overview of Natural Language Understanding (NLU) technologies. Concept Error Rate (CER) metric and Concept Level Confidence metrics. “The syntax of a language is seen as algebra and grammatical categories are seen as functions”

Degen, Leo. "Working with audio: integrating personal tape recorders and desktop computers." ACM Conference on Human Factors in Computing Systems - CHI '92 (1992):413-418.

Annotations: Uses buttons to mark audio start and end of interesting segments.

Ernst, R. 2001, "Combining speech recognition software with digital imaging and communications in medicine (DICOM) workstation software on a microsoft windows platform", Journal of Digital Imaging, vol. 14, no. 2, pp. 182.

Annotations: Radiology application, using Power Scribe (post-cursor of the Voice Navigator). Results showed near 100% dictation speech recognition accuracy. Most common mistake was “or” instead of “for”, in a complex domain-specific sentence. Also used as a reference in the Technical Report.

Feng, J. 2004, "Using confidence scores to improve hands-free speech based navigation in continuous dictation systems", *ACM transactions on computer-human interaction*, vol. 11, no. 4, pp. 329.

Annotations: Confidence scores range from +30 down to -15. Navigation and selecting incorrect words for correction is a primary focus.

Gorniak, P. 2003, "Augmenting user interfaces with adaptive speech commands", *ICMI'03: Fifth International Conference on Multimodal Interfaces*, pp. 176.

Annotations: Teach the computer what each tool is called by clicking on it, and saying the name. Future work on “SayPlay” could allow users to define synonyms for commands by using a Meta command such as: “Computer, redefine the command X to {instead or also} be spoken as Y, {instead or also}”. Where X is an existing command (single word or phrase, such as “Skip to Start”, and Y is the new command phrase, such as “Jump to the beginning”. This helps users recall the precise phrases for each command.

The challenge implementing it is to use X to get the Semantic Result for that command (perhaps in a table lookup), and to assign the semantic result of Y, so both phrases result in the same command.

Gruenstein, A. 2008, "The WAMI toolkit for developing, deploying, and evaluating Web-Accessible multimodal interfaces", *ICMI'08: Proceedings of the 10th International Conference on Multimodal Interfaces*, pp. 141.

Annotations: Detailed description of software architectural structures useful in providing hands-free software user interfaces. Most commands are combinations

of gestures and spoken commands, such as drawing lines or pointing to objects, naming them and performing some action upon them.

Hanna, P., O'Neill, I., Wootton, C. & McTear, M. 2007, "Promoting extension and reuse in a spoken dialog manager: An evaluation of the queen's communicator", *ACM Transactions on Speech and Language Processing*, vol. 4, no. 3.

Annotations: Good coverage of object oriented design principles as applied to a spoken dialog manager. Some notable advantages are a design that can be extended to handle unforeseen situations such as: all inquiry agents inherit a common core of discourse management behavior, thereby ensuring all agents employ a similar discourse style. A domain spotter object is used to manage interactions between the user and agents. Finally, a shared discourse history is used to provide an evolving account of the dialog. Page 10.

Hartman, M.T. 1998, "Talking to your television", Proceedings of 42nd Annual Meeting of the Human Factors and Ergonomics Society, p. 532.

Annotations: Experiments with user behaviors and satisfaction using a voice command controlled television remote device. The device itself was a mockup, made to appear to be a truly functional device, with a "Man Behind the Curtain" performing the actual recognition. The goal was to simulate a real device in order to measure levels of dissatisfaction caused by recognition errors, which were introduced based on a random function. The results were universally in favor of having such a device.

Hubbell, T.J., Langan, D.D. & Hain, T.F. 2006, "A voice-activated syntax-directed editor for manually disabled programmers", *Eighth International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2006, October 23, 2006 - October 25* Association for Computing Machinery, Portland, OR, United states, pp. 205.

Annotations: The way to Name things is using a recording paradigm to start, record the name, and stop when finished. The user can correct mistakes

using “back” and restating these parts. Test methodology used 5-point Likert test questionnaire of several programmers.

Quotation: “The Name control allows users to create new identifier names for projects, classes, methods, fields, and variables, and is one of the few places that employs the *dictation grammar of the recognizer*.” P. 208

Iwahashi, N. "Interactive learning of spoken words and their meanings through an audio-visual interface." *IEICE Transactions on Information and Systems* 91.2 (2008):312-21.

Annotations: Teaching words interactively based on computer system queries, such as asking the user “what is this word?” whenever a new word is encountered. The user provides visual and keyboard feedback.

Jiang, H. 2005, "Confidence measures for speech recognition: A survey", *Speech Communication*, vol. 45, no. 4, pp. 455.

Annotations: Confidence Measure (CM) as used in practical applications (whether or not to issue a command based on a recognized utterance). Also, CM versus Utterance Verification (UV), meaning whether a sound is in fact a spoken utterance.

Ke uml puska, V.Z. 2009, "A novel Wake-Up-Word speech recognition system, Wake-Up-Word recognition task, technology and evaluation", *Nonlinear analysis*, vol. 71, no. 12, pp. 2772.

Annotations: Wake-up-words are distinguished from keyword spotting by distinguishing between words spoken to the computer, and words spoken to others about the computer. This is done without natural language understanding models.

Keefer, R. 2010, "Voice commands for a mobile reading device for the visually impaired", *ACM International Conference Proceeding Series*

Annotations: Detailed descriptions of grammars for providing flexible command phrases. Selecting sections of newsprint for playback is conceptually similar to selecting sections of audio to playback.

Klarlund, N. (2003) "Editing by voice and the role of sequential symbol systems for improved human-to-computer information rates" Proceedings of International Conference on Acoustics, Speech and Signal Processing ICASSP '03 pp. 728-31

Annotations: Describes ShortTalk, a useful language designed for human computer interaction by voice.

"Two principles for realizing this goal for dictation systems: srenophonic concept naming-fundamental concepts are treated as symbols that receive short pronunciations- and unambiguous orthogonality-the command grammar must allow concepts to be sequenced combinatorially with few restrictions while preserving distinctness of commands from fragments of natural language." p.V-728.

The idea of concept naming is similar to the entity naming of tracks for purposes of issuing commands on them.

Kou, X.Y. 2010, "Knowledge-guided inference for voice-enabled CAD", *Computer aided design*, vol. 42, no. 6, pp. 545.

Annotations: Utilizes Hypernyms and Hyponyms, words that are more or less general than a given word.

Kim, Ji-Hwan 2010 "Transformation-based named entity extraction from spoken content for personal memory aid" (Dept. of Computer Science and Engineering, Sogang University, Seoul, Korea, Republic of) Source: IEEE Transactions on Consumer Electronics, v 56, n 4, p 2606-2614, November 2010

Annotations: Named entities (NE) are generally proper nouns. Spoken dialog does not have the advantage of using Capitalization, as in text. This paper

compares its rule-based NE extraction with the best-performing commercially available stochastic NE system. 2 steps: preprocessing, automatic rule generation
Quote from page 2607:

“Speech disfluencies such as filled pauses and repetitions are prevalent in spontaneous speech. Unlike the corruption of input, these kinds of error do not come from speech recogniser errors but from the disfluencies themselves. In these cases of disfluency, any missing elements or extra intervening tokens can cause mismatches between trained patterns and input speech. Speech disfluencies are classified as Filled pauses (e.g. Cambridge uh * university), Repetitions (e.g. Johnson * Johnson was here), Repairs (e.g. Johnson * Jackson liked it)”.

Koo, M., Choi, J. & Kim, Y. 2008, "The development of automatic speech recognition software for portable devices", 1st International Conference on Advances in Computer-Human Interaction, ACHI 2008, Feb. 10-15, 2008 IEEE Computer Society, Saint Luce, Martinique, pp. 59.

Annotations: Concerned with Utterance Verification (UV), in hand-held devices for medium to close mic. Compare 2 popular smart phones with voice commands HP and Samsung.

Langer, S., 2002 “Impact of Speech Recognition on Radiologist Productivity” Journal of Digital Imaging, Vol 15, No 4, pp 203-209

Annotations: Speech Recognition reduced the time required to produce a report and increased the number of reports per day for radiologists.

Lemon, O., Gruenstein, A., 2004. "Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments." ACM transactions on computer-human interaction 11.3 pp.241-267.

Annotations: Conversations are by nature multi-threaded, meaning that several lines of thought can be going on simultaneously. Context sensitivity limits the possible referents to within a context. Corrective Fragments refer to the user making a correction to the current actions by issuing a revised command.

Lee, K.-F. 1988 "Large-vocabulary speaker-independent continuous speech recognition- The SPHINX system" PhD. Thesis, Carnegie-Mellon University

Annotations: Highly referenced source for successful Dictation speech recognition systems. The strategy was to attack both Large Vocabulary and Continuous (rather than keyword) speech recognition for dictation recognition systems. Speaker independence was also addressed, but with less success without some training, than given some amount of training.

Lepinski, G.J. & Vertegaal, R. 2008, "Using eye contact and contextual speech recognition for hands-free surgical charting", 2nd International Conference on Pervasive Computing Technologies for Healthcare 2008, Pervasive Health, January 30,2008 - February 01 IEEE Computer Society, Tampere, Finland, pp. 119.

Annotations: Gaze Recognition is used to trigger a gate which allows Voice Commands to be understood as actual commands, and not regular speech. Results show a drastic improvement over using Speech Recognition without the gaze detection to govern which speech is to be recognized as a command. Also see the reference: L. Rossi, D. Sacerdoti, B. Billi, G. Lesnoni, M. Orciuolo, T. Rossi, D. Sacerdoti, and L. Bertollini, "Automatic speech recognition in vitreoretinal surgery. A project for a prototypal computer-based voice controlled vitrectomy machine," European journal of ophthalmology vol. 6, no. 4, pp. 454-459, 1996.

Robotics: Note there are several other Voice Command Robotics examples: Voice controlled teleoperated Robot, Voice Command for Robot Guidance, HTK for Voice controlled robotic system based on Hidden Markov Model, Development of a Cognitive Model of Humans in a Multi-Agent Framework for Human-Robot Interaction

Lopez-Cozar, R. 2006, "Testing the performance of spoken dialogue systems by means of an artificially simulated user", The Artificial Intelligence Review, vol. 26, no. 4, pp. 291-323.

Annotations: Discusses the implementation of a simulated user for analysis of all of the various paths through a menu driven dialog system. Simulated users do not get frustrated or tired, and the variance of an experiment can be narrowed. Also, a great many more sequences can be tested.

McCauley, Lee & D’Mello, Sidney & Daily, Steve 2005 “Understanding without Formality: Augmenting Speech Recognition to Understand Informal Verbal Commands”, *43rd ACM Southeast Conference March 18-20, 2005 –Kennesaw, Georgia USA*, pp. 1.42–7

Annotations: Uses Dictation Speech Recognition to generate a listing which is analyzed offline by a process called Latent Semantic Analysis (LSA). Simultaneous to this, the command grammar based (key-phrase) speech recognition is used to interact with the speaker. Grammars are updated when the LSA returns a new phrase for an equivalent command.

Nakano, Teppei 2008 “Flexible Shortcuts: Designing a New Speech User Interface for Command Execution”, *CHI 2008 Proceedings, April 4-8, 2008 – Florence Italy*, pp. 2621-2624.

Annotations: Continuous Keyword Input allows the user to be more flexible in each spoken command. Results compare well against traditional Command and Control (C&C). Experiment compares control of Windows Media Player and Voice Chat (Skype), and is measured using subjective assessment.

Nolan, B 2002, "The JAM Suite: A Voice-Enabled Network-Based Virtual Band Application". *Principles and Practice of Programming in JAVA*.

Annotations: There are no measurements and conclusions to lead one to believe this system was actually implemented. Disparate areas of Pitch Detection, multipoint network streaming, and speech recognition for voice commands are described, but no results are discussed. It seems like a nice idea but is entirely impractical due to latency problems, as discussed.

Motorola MOTOROKR E8: Voice Command Instructions, [http://support.t-mobile.com/doc/tm52272.xml?related=y&Referring Related DocID List Index=3&navtypeid=6&pagetypeid=7&prevPageIndex=1](http://support.t-mobile.com/doc/tm52272.xml?related=y&Referring%20DocID%20List%20Index=3&navtypeid=6&pagetypeid=7&prevPageIndex=1)

Annotations:Names (first and last) of contacts are stored, to call back by voice commands.

Dial number, Send message to <contact’s name>, Check calendar, Check new message, Check new e-mail, Add new contact, Talking phone, Check

battery, Check signal, Check time, Check date, Open setup, Open recent calls, Open theme, Open camera, Open Web access, Set normal, Set vibrate, Set silent, Set ring, Set airplane, Set airplane off

Obuchi, Y. 2010, "Intentional voice command detection for trigger-free speech interface", IEICE Transactions on Information and Systems, vol. 93, no. 9, pp. 2440.

Annotations: Voice Activity Detection, can be key to minimizing dysfluency misrecognitions.

Rebman, C M. "Speech recognition in the human-computer interface." Information & management 40.6 (2003):509-19.

Annotations: Provides an overview of different classes of SR, including speaker dependent versus speaker independent, continuous versus discrete. Performance measurements are primarily concerned with dictation speech recognition accuracy and rates for time-savings. It also provides a good model of presenting results.

Ross S., Brownholtz E., Armes, R., 2004 "Voice User Interface Principles for a Conversational Agent" IUI'04, Jan. 13–16, 2004, Madeira, Funchal, Portugal. ACM

Annotations: These ideas serve as organizing principles for conversational systems, such as HAL, or the computer on Star Trek. These require specific behaviors for interruption, acknowledgement, and other cases of interaction. The one-to-one nature of request and response has been improved upon in Lemon and Gruenstein, 2004.

Rudnicky, Alexander I. 1989 "*The Design of Voice-Driven Interfaces*", Human Language Technology Conference archive. Proceedings of the workshop on Speech and Natural Language. Association for Computational Linguistics Morristown, NJ, USA 120 - 124

Annotations: Design the Language of the Command Set based on observations of users performing tasks by voice in unconstrained situations. Design facilities to “promote fluent interaction, error repair, and capability to introduce new (task-specific) words”. Voice controlled Spreadsheet application.

Russell, Norvig, Peter; “Artificial Intelligence: A Modern Approach”

Annotations: Textbook used to teach Artificial Intelligence at graduate or undergraduate level. Describes agent based systems in detail.

Saruwatari, H. 2009, "Hands-free speech recognition challenge for real-world speech dialogue systems", ICASSP 2009 - 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3729-32.

Annotations: Real-world acoustical issues, such as ambient noise and reverberation, are addressed using a microphone array. A noise estimation experiment is performed, using the Blind Spatial Subtraction Array (BSSA) and results show improvements over other microphone techniques.

Schmandt, Christopher & Hulteen, Eric A., “The intelligent voice-interactive interface”, Proceedings of the 1982 conference on Human factors in computing systems, p.363-366, March 15-17, 1982, Gaithersburg, Maryland, United States

Annotations: The gesture and voice user interface developed and described here is succinctly called “Put that there”. A simple user interface to build and modify a database is described wherein gestures and spoken commands are combined to control the interaction with the database. Ambiguities are resolved by synthetic spoken questions for clarification (such as asking “Which one?”)

Schmandt, Chris, Ackerman, Mark S., Hindus, Debby. 1990, "Augmenting a Window System with Speech Input", IEEE Computer Magazine, pp. 50-56.

Annotations: Uses XSpeak II as voice recognition technology to allow users to switch between windows in a graphical user interface. p176.

In this paper, we propose an adaptive speech input augmentation

to any standard graphical user interface. With this speech layer added to the interface, the user may continue to use the system as before, controlled solely with keyboard and mouse. However, he or she can also use speech to name interface actions such as button clicks. A phoneme recognizer produces a phonetic representation of the utterance, which is associated with the interface action that occurs closest in time. After a few consistent examples, the user can speak the name he or she chooses for the action instead of using the mouse or keyboard.

Schuricht, M. 2009, "Managing multiple speech-enabled applications in a mobile handheld device", *International Journal of Pervasive Computing and Communications*, vol. 5, no. 3, pp. 332.

Annotations: Prototype uses DynaSpeak from SRI, running under Windows XP. DynaSpeak includes noise cancellation, endpoint control, dynamic grammar update, and easy API. Program Manager controls the flow of information between applications, based on voice command queries.

"there are several individuals that have the same first name, or names that are similar in pronunciation. Thus, the program must differentiate first name requests by asking the user for more information about the person to whom they are referring.

Sepe Jr., R.B. & Pace, J.F. 1999, "Voice actuation with context learning for intelligent machine control", *Proceedings of the 1999 IEEE Industry Applications Conference - 34th IAS Annual Meeting, October 03, 1999 - October 07* IEEE, Phoenix, AZ, USA, pp. 295.

Annotations: Predicting the next likely action based on previous patterns of activity, recommends the next action when user seems stuck, and requests confirmation whenever an unlikely sequence of commands are used. A discrete Markov model is used to develop a statistical model of usage, which then drives the likelihood feedback provided for new users to learn the process, and experienced users avoid errors in process control.

Sirota, Milton, 1965, "Minimum Sample Sizes For Superiority Comparisons of Prior Tested Items", *Industrial Quality Control*, June 1965, pp. 603-605.

Annotations: Probabilistic and statistical model for calculating the length of a string of successful tests necessary to consider an improvement to be statistically significant, given test results from prior to making the modification which is being considered as a possible improvement. The equation from this paper was implemented into a spreadsheet, allowing any number of past measured success/total ratios to be used in determining the sample size necessary to produce a 95% confidence measure in test results taken after making a modification.

Stifelman, L.J., Arons, B., Schmandt, C. and Hulteen, E.A. "VoiceNotes: A speech interface for a hand-held voice notetaker". In Proceedings. CHI93 (1993), pp.179-186

Annotations: Capturing and retrieving spontaneous ideas, recorded as spoken voice. Main techniques are snap to start, and varispeed playback.

Suendermann, D., Liscombe, J., Dayanidhi, K., Pieraccini, R., 2009, "A Handsome Set of Metrics to Measure Utterance Classification Performance in Spoken Dialog Systems" Proceedings of SIGDIAL 2009: the 10th Annual Meeting of the Special Interest Group in Discourse and Dialogue, pages 349–356

Annotations: Caller experience measurement methods. Compare "True Total" versus Caller Experience as basis for accurate measurement.

Troutman, John F., Miller, Joy A.; "Homophones List"
<http://fivejs.wordpress.com>, accessed 5/5/11.

Annotations: This list of 2000 Homophone pairs was filtered down to a set of 50 homophone pairs that could have anything to do with music recording, such as Lyre/Liar. These were used as input for an experiment to measure the effectiveness of Elaboration in tipping the balance of recognition toward one spelling or the other.

Yavelow, Peter 1989, "Voice Navigation for the Macintosh Musician". Articulate Systems marketing literature

Annotations: Voice commands to aid in music production, calling up patches by name, as well as remote control.

Yegulalp, Serdar 2011; “Speech recognition: Your smartphone gets smarter”
http://www.computerworld.com/s/article/9213925/Speech_recognition_Your_smartphone_gets_smarter?taxonomyId=15&pageNumber=1

Annotations: Good description of Statistical Language Model used in speech recognition.

Zhou, H., Sadka, A. & Jiang, R.M. 2008, "Feature extraction for speech and music discrimination", 2008 International Workshop on Content-based Multimedia Indexing - CBMI 2008 IEEE, Piscataway, NJ, USA, pp. 170.

Annotations: Tutorial on differentiating speech signals from music signals, at the spectrographic level. If this technology could differentiate singing from speaking, then “Stop in the name of love”, might not halt recording. A simpler way is to require the user to say “Stop Recording” when in record.

APPENDIX I: GLOSSARY

Dictation Speech Recognition: Unconstrained, free-running language capture and render as text. The other kind of speech recognition is “Key word spotting” recognition, where specific phrases are codified and recognition events are triggered upon the detection of the utterance of these phrases. Dictation speech recognition is also called Continuous Speech Recognition.

Disfluency: An utterance which is not a word, nor is intended to be recognized as a word, such as clearing one’s throat, licking one’s lips, stammering, or other accidental sound. These are generally associated with an accidental or erroneous recognition event.

Homophone: A word associated with another word that sounds the same when spoken, but has different meaning and spelling.

Homograph: A word associated with another word that is spelled the same way, but is pronounced differently, and has different meaning.

Keyword Speech Recognition: Recognition of words and phrases that are predefined and programmed into a grammar structure which is loaded into the speech recognition engine.

Overdub: Adding a new recorded track alongside existing recorded tracks in a multi-track recording session. Historically, the number of tracks in a recording was limited to the tape recorder. The number of tracks available increased from 4 tracks in the mid 1960’s to 8 tracks in the late 1960’s to 16, 24 and 32 tracks into the 1980’s, with commensurate increases in system cost. With digital audio recording software, the number of tracks is limited by the disk access speed and disk throughput. For an average laptop computer with a single disk drive in 2011, the number of tracks that can possibly be played back at one time, while recording a new track is about 32 tracks.

Recognition Event: An event initiated by the Speech Recognition Engine as a result of the recognition of an utterance into the microphone by the user.

Session: A “Session” is a meeting for collaboration in the task of recording a series of musical pieces. It generally takes place in a recording studio or concert hall where acoustics are appropriate to good sound reproduction.

Slot: A “Slot” is a place within a grammatical structure where a set or selection of words may occur, such as the names for ingredients to include on a pizza in the example: “I would like to order a large pizza with pepperoni, mushrooms and sausage.” The word “Large Pizza” fills a slot for what is being ordered, and pepperoni, mushrooms and sausage” fill slots for extra ingredients. In “SayPlay” the primary slot is for the track name in the track assignment commands: “Name this track X”, and “Name the recorded track X” where X represents the slot for the name.

Take: A “Take” is an audio recording of a run-through of the audio events being performed. For example when an ensemble performs a piece that is being recorded, the “take” consists of all recorded events from the start of recording until recording is halted. A take can be aborted and thus becomes an aborted take.

Track: A Track is a continuous linear recording of an audio signal. A track is a single channel, from a microphone into an input and then into a file stored on the computer hard disk drive. It can be multiple channels, as when supplying several microphones to each individual drum and cymbal in a drum set. In this case the collection of tracks comprising the drums can be referred to as the drum tracks, but more commonly each channel will still be referred to by the drum, such as the snare drum track, or the kick drum track.

Multi-track recordings involve discrete channels for signals from microphone to record media, such that they can be played back in isolation from the recordings of the other instruments in the ensemble

VITA

NAME OF AUTHOR: John Martin Goddard

PLACE OF BIRTH: Denver, Colorado, USA

DATE OF BIRTH: November 12, 1965

UNDERGRADUATE SCHOOL ATTENDED:

University of Colorado at Boulder

DEGREES AWARDED: Bachelor of Science in Electrical and Computer Engineering,
Bachelor of Arts in Music, 1989, University of Colorado at Boulder

PROFESSIONAL EXPERIENCE:

Digital Audio Programmer, Sonic Solutions, San Francisco CA 1990-1992

Software Engineer, Digidesign, Palo Alto CA, 1993-1996

Licensing Standards and Test Development Engineer, Dolby Laboratories
Licensing Corporation, San Francisco CA, 1997-2003

Senior Sales Engineer, Wohler Technologies, Hayward CA, 2004-2006

Senior Applications Engineer, Telairity Inc., Santa Clara CA 2006-2007

HONORS AND AWARDS:

Nomination for Daytime Emmy Award, Best Sound Editing, Disney's
Alladin 1995