Enabling Non-Speech Experts to Develop Usable Speech-User Interfaces

Anuj Kumar

CMU-HCII-14-105

August 2014

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Committee:

Florian Metze	Co-Chair, Carnegie Mellon University
Matthew Kam	Co-Chair, Carnegie Mellon University & American Institutes for Research
Dan Siewiorek	Carnegie Mellon University
Tim Paek	Microsoft Research

The research described in this dissertation was supported by the National Science Foundation under grants IIS-1247368 and CNS-1205589, Siebel Scholar Fellowship 2014, Carnegie Mellon's Human-Computer Interaction Institute, and Nokia. Any findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the above organizations or corporations.

KEYWORDS

Human-Computer Interaction, Machine Learning, Non-Experts, Rapid Prototyping, Speech-User Interfaces, Speech Recognition, Toolkit Development.

Dedicated to Mom and Dad for their
eternal support, motivation, and love

ABSTRACT

Speech user interfaces (SUIs) such as Apple's Siri, Microsoft's Cortana, and Google Now are becoming increasingly popular. However, despite years of research, such interfaces really only work for specific users, such as adult native speakers of English, when in fact, many other users such as non-native speakers or children stand to benefit at least as much, if not more. The problem in developing SUIs for such users or for other acoustic or language situations is the expertise, time, and cost in building an initial system that works reasonably well, and can be deployed to collect more data or also to establish a group of loyal users. In particular, application developers or researchers who are not speech experts find it excruciatingly difficult to build a testable speech interface on their own, and instead routinely resort to Wizard-of-Oz experiments.

To address the above problem, we take the view that while it can take prohibitive amount of time and cost to train non-experts into the nuances of speech recognition and user-interface development, well-trained speech experts and user-interface specialists who routinely build working recognizers have accumulated years of experiential knowledge that we can study and formalize for the benefit of non-experts. As such, the core speech recognition technology has reached a point where given enough expertise and in-domain data, a working system can be developed for almost every user group, acoustic or language situation. To this end, we design, develop, and evaluate a speech toolkit called SToNE, which embeds expert knowledge and lowers the entry bar for non-experts into the design and development space of speech systems. Our goal is not to render the speech expert superfluous, but to make it easier for non-speech experts to figure out why a speech system is failing, and guide their efforts in the right direction.

We investigate three research goals: (i) *how* can we elicit and formalize the tacit knowledge that speech experts employ in building an accurate recognizer, (ii) *what* are the different analysis supports – automatic or semi-automatic – that we can develop to enable speech recognizer development by non-experts, and (iii) *to what extent* do non-experts benefit from SToNE. Through experiments both in the lab with new datasets, and summative evaluations with non-experts, we show that with the support of SToNE, non-experts are able to build recognizers with accuracy similar to that of experts, as well as achieve significant gains from when SToNE support is unavailable to them.

This work aims to support the "black art" in SUI development. It contributes to human-computer interaction by developing tools that support non-speech experts in building usable SUIs. It also contributes to speech technologies by formalizing experts knowledge and offering a set of tools to analyze speech data systematically.

ACKNOWLEDGEMENTS

A PhD is not just a destination, it is a journey. It is nearly impossible to enumerate and write about the people who have made this journey both intellectually stimulating and fun for me, as any attempt will be incomplete and any words will be far too less. Nonetheless, here's my attempt. It is the very least I can do.

Research cannot happen in a vacuum – unless of course, if you are an experimental astrophysicist. A number of amazing people across academia and industry have played a huge role in shaping up my experience as a PhD researcher, and I have been immensely privileged to be around them.

First and foremost, I have had the great luck of being advised by two remarkable researchers: Florian Metze and Matthew Kam. Under their wings, I have learnt how to be passionate about a research topic, how to think (or not to) when you are stuck, how to break down an insurmountable problem into smaller pieces, and how to be relevant and aim big. I would like to thank both of them for supporting me throughout my time at CMU, no matter what.

Florian Metze and I go back to my early days as a PhD student when I took his course on "Speech Recognition and Understanding." It was this course that got me interested in Speech Recognition, so much so, that I decided to pursue it more seriously in my PhD thesis with him. Florian has a rare gift among advisors: the ability to delicately balance a hands-on/ hands-off approach and lead by example. This allows his students to thrive, to never fall back, to always have someone to look up to, and to be able to slowly nurture their own paths that they enjoy. This is what makes the PhD journey both exciting and fruitful. It makes the students who work with him feel like they are working with a true collaborator, instead of a boss. He also has a great technical repertoire across multiple fields, which always gave me a sense of confidence to pop in his office and ask any question under the sun. I always knew that whatever be the question, Florian might have an answer – or if not something definite, at least a direction to think in. Florian has been a great advisor and a friend to brainstorm many research ideas with. He has the ability of taking a topic, fleshing it out, and making others in the room enthusiastic about it. As a result, I have not only been able to work on my own thesis, but also discuss and submit a number of federal and industry grant proposals with him, many of which were awarded. It is these qualities that I admire most. He has enthusiastically supported me throughout my time at CMU, and for that I will always be indebted.

Matthew Kam and I go way back to my sophomore days as an undergraduate student in India. He was first my undergraduate mentor and later, my PhD thesis advisor at CMU. It is because of Matt that I got excited about research, understood what high-quality research is all about, and got trained on how to properly conduct a research agenda. Matt inspired me to pursue

graduate studies then, and had it not been his guidance during my undergraduate days, I would not be adequately prepared for graduate school. I still remember my early days as an undergraduate researcher when I had no understanding of what research meant, and considered writing somewhat functional code as research. From that I went to being confident of thinking about my own research agenda, leading small research groups including multi-disciplinary participants, writing my own top-tier publications, and presenting my work to a large audience. Matt has taught me to look at the big picture and to take risks. I still remember his words about having a "helicopter vision," which is often necessary to stay focused. Matt invests in people, rather than ideas. He invested in me, allowed me to explore ideas in my early days as a PhD student, and encouraged me all the way. He won't let his students sell themselves short, and heavily protects their interests, even if it means fighting tooth and nail for them. This is a profoundly motivating and supportive method of advising, for which I am incredibly thankful.

I would also like to thank my other committee members: Dan Siewiorek and Tim Paek. Both of them have been a source of great support and imparted sage advice over years. Dan and I met somewhat in the middle of my PhD career when I took his class on "Mobile and Pervasive Computing," and I was instantly awestruck by his sheer technical brilliance coupled with decades of experience. He had a number of war stories to narrate ranging from "why things will nearly always go wrong when you don't want them to, and how to prepare for it," to "how you should thrive in a multidisciplinary environment." It these short – but extremely interesting lessons – that made him a perfect committee member, and I have always felt very comfortable bouncing ideas off of him, and getting his opinion on research. For this support, I am very grateful. Tim Paek and I met at a workshop on Speech Recognition and Human-Computer Interaction, which he was organizing, and immediately after that I went on to intern with him at Microsoft Research for the summer. It was one of the most fascinating three months of my research life. Tim proves that you can be both a highly successful researcher and the nicest person in the world. His focus on thinking about research both from a scientific standpoint, as well as, an industry standpoint still stands out. It is his words of encouragement and thinking that probably has stuck till date, and encouraged me to pursue professional life in industry post-graduation. Tim has also been a great support for the thesis – his experience both in HCI and Speech Recognition, made him the quintessential committee member, and I was inspired by his depth of technical acumen in both fields simultaneously. I am filled with gratitude for him for providing an opportunity at Microsoft Research early on, for being an ever so enthusiastic supporter of my PhD work thereafter, and for being so friendly and full of energy in any and every interaction we have had in the past few years.

Most people tell you that your peers – the people in your research group – are more instrumental in shaping your personality and experience than your advisors. That is true. Being around very smart people makes you smarter, and as a part of two talented research groups, I

have had the privilege of witnessing this first hand. In Florian's group, I found a group of students eager to build the best of the speech recognition systems. I'd like to thank Yajie Miao and Ankur Gandhe for their constant support and willingness to discuss at length any technical questions thrown their way on speech systems; Udhaykumar Nallaswamy for being a great mentor early on; and Tina Milo, Lucas Czech, Lara Martin, Hao Zhang for being great buddies. Special mention for Nikolas Wolfe who has been a great hangout friend – his jokes and ever so lively approach to life has what kept the last few months of PhD life interesting. Also, special thanks to Eric Riebling, who has been a great officemate, and a great friend.

Matt's research group was a slightly different beast. As an "older" researcher in the group, I have had the pleasure of interacting with and mentoring many younger students that I saw enter and grow. This experience is what taught me that mentoring is tougher than you think, and that advisors – like mine – who have done an excellent job at mentoring deserve total credit for it. An all round thank you to Indrani Vedula for helping setup the logistics of field studies, and Rajat Agarwal, Kaushik Shelat and Venkatesh Keri for helping execute them. The field studies in India would have been a totally different ball game without your support.

There are have been numerous other PhD students who have been instrumental in guiding me through the years. Tawanna Dillahunt and Brian Lim have probably been the biggest support in my early years as a PhD student. I have hounded them with smallest of questions, and they have very patiently and kindly answered all of them. Life would have been tough without their support. Eiji Hayashi and Derek Lomas have been great friends as well. Eiji and I were roommates during the Microsoft Internship and both of us sailed through some tough but happy times during the summer together. Derek and I have been the best of research mates for the first two years before I decided to jump onto speech recognition (vs. educational games). We had complimentary skill sets, which made the whole collaborative experience even more fabulous. Technically, we were in the same cohort, but in terms of life experiences and ability to deal with stress ranging from giving presentations to thinking holistically, he has taught me quite a bit. Thank you everyone for your support and kindness.

A big round of applause for all my research collaborators. I have enjoyed tackling hard problems, sharing ideas, and building solutions with you. Especially Anuj Tewari, Pooja Reddy, Siddharth Kothari, Malav Bhavsar, and Yashesh Gaur, thank you so much for working with me. Anuj Tewari and I go back to the days of undergraduate college – he was a year senior to me, and I always found a great example in him for almost anything. Looking at him, I found peace in the fact that the path that I am following is going to lead somewhere good – and had it not been because of him as an example I might have not decided to pursue a PhD myself! Pooja Reddy and I worked for the first few years of my PhD together, and it was her who taught me an entirely new field of literacy. She is probably one of the smartest and kindest PhD students I

have come across, and without her support, the educational games project probably wouldn't have been possible. Thank you so much for working and collaborating with me – all of you!

I have had the opportunity of two fantastic internships, one at Microsoft Research (Redmond) with Tim Paek and Bongshin Lee, and the other at IBM Research (Almaden) with Barton Smith and Chris Kau. Thank you all for working with me, and showing me the ropes of industry life. These experiences have been the single most influential factor in my decision to pursue a career in industry upon graduation.

Any CMU PhD life is incomplete without the fabulous support from the administrative staff. Queenie Kravitz, Jo Bodnar, Faith Boldt, and Jae Cho – thank you so much for supporting my stay at CMU. I have been amazed how each time I threw a tough question your way, you found a way – by even sometimes going outside the norm – to answer and resolve it. This PhD would have been a different story without your nourishing support.

In terms of friends, the list is really long, and I will inevitably miss a few. If I do forget, I apologize. First, I have been lucky to find friends in school and undergraduate days that have stuck on, despite being not co-located. Laveen Ramrakhiyani, Satyajit Swain, Sagun Garg, Puneet Maheshwari, and Apurv Agarwal deserve a special mention here. It's them that I have called up to first share the best of news, and it's them that I have called to discuss my problems. I am thankful to have such a great support net in them, no matter where I go.

My CMU and Pittsburgh experience has been largely shaped by a group of people that has stuck very closely with each other through ups and downs. I would like to thank Ankit Sharma for being the best of apartment mates, for challenging my beliefs every now and then, for all the lunch and dinner discussions, and for the pre-, on-time, and post-birthday celebrations; Mehdi Samadi for being ever so cheerful, and for the hilarious moments of confusion and inspirational moments of kindness; Anvesh Komuravelli for being inhumanely easy going and providing entertainment and perspective in life; Pallav Vyas for providing a listening ear whenever in need; and Ashique Khudabukhsh for all the tennis challenges. I would also like to thank Shayak Sen, Shrikant Adhikarla, and Mayur Sharma for making my time at CMU worth cherishing. My workout buddies Sriram Somanchi and Alessandro Antonsambetta also deserve a special mention here. Not only did the workouts with them keep me sane and fit, but they also provided me with some amazing memories. I am looking forward to regrouping with all of you on the West Coast.

None of this research would have been possible without the generous funding support from numerous agencies. I'd like to thank the National Science Foundation for funding through grants IIS-1247368 "EAGER: A Research Infrastructure for Analyzing Speech-based Interfaces," and CNS-1205589 "The Speech Recognition Virtual Kitchen." I have also been the beneficiary of a generous fellowship from Siebel Foundation for my last year of PhD studies, as also support from CMU's HCII department through Matthew Kam's faculty startup funds. Nokia

also provided support for purchasing research equipment and conference travel. Thank you so much for providing financial support without which this PhD would not have been possible.

Finally, I would like to express my utmost debt and gratitude to my parents to whom I have dedicated this thesis. Coming from India, opting for a PhD is a difficult career decision – as it's hard to explain the merit and worth of a PhD vs. equivalent time in the industry. However, both Mom and Dad not only supported me whole-heartedly but also inspired me to go for it. I still remember a time when I doubted my own decision of pursuing the program, but it was dad's words of motivation and mom's words of comfort in the career choice that I had made, which made me go through with it. Today, I am a mere reflection of their teachings, and it is their message of perseverance, never give up attitude, and sincerity that has carried me through PhD. As my sisters put it, dad is the "genie of the house" who solves any and every difficulty, and mom is the "doctor of the house" who will not only nurse you when you are ill, but also make you forget any problem you may have, as she will make it her own until it is resolved. It is this level of comfort that has put me at ease during the long, testing tenure of PhD that whatever be the issue, it will be taken care of. It has been the vital ingredient in keeping me focused and determined throughout. Thank you Mumma and Papa!

My other family members have been equally caring. Both my elder sisters – Sonika Gupta and Priyanka Kumar – have supported me through thick and thin, and it is their love that has mattered the most at times when results were hard to come by in research. Sonika, my eldest sister, has been the strong backbone of the family, and it is her who I fall back to whenever I feel too muzzled up with small details for gaining the big picture. Priyanka, my other elder sister, and I have had umpteen discussions ranging from my career trajectory to what not, and she has always been the one to show me the way forward. Thank you to both my brother-in-laws as well – Ashish Gupta and Gaurav Singh. Ashish was the one who had inspired me to pick CMU (over other competitive schools), and had it not been because of him, I might not have been at this wonderful place! Gaurav, on the other hand, had taught me another important life lesson early on in my undergraduate days: to apply my brain for simple tasks (instead of depending on Google search), and had it not been for that one profound lesson, my thinking prowess would have been entirely different. Thank you also to my nephew, Aditya for teaching me that life need not be complicated, and that major achievements are possible with the simplest of approaches in life. A final round of gratitude for my grandparents without whose love and support life till date would have been a different story. Thank you everyone!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Speech-user interfaces (SUIs) are becoming increasingly popular, both in the developing and the developed world. For users in developing regions, where illiteracy often impedes usage of graphical interfaces, spoken dialog systems are being explored to provide access to relevant information, such as weather information to farmers [Plauchè *et al.*, 2006], medical advise to women [Sherwani *et al.*, 2009], and also in many applications of entertainment services [Raza *et al.*, 2013]. Similarly, several applications of speech interfaces have gained momentum in the developed world. For instance, United Airlines' customer care system, which receives an average of two million calls per month, indicated that their caller abandonment dropped by 50% after implementing a speech system to replace a touchtone system [Kotelly, 2003]. Such a solution introduced eyes-free, hands-free operation along with ease of data entry for non-numeric text [Hura, 2008]. Recently introduced mobile assistants like Apple's Siri [apple-siri], Microsoft's Cortana [microsoft-cortana], and Google Now [google-now] have further boosted the popularity of speech applications. They assist in executing simple commands such as "Call John", "Text my wife that I'll be late today", or "Set a wakeup alarm for tomorrow 6am!" to more complicated tasks such as dictation or transcribing a phone conversation. These examples underscore the reasons why many researchers and practitioners are becoming increasingly interested in incorporating speech as an input and interaction modality in their applications.

Yet, it is telling that beyond a few examples, we haven't seen a large number of functional and sufficiently accurate speech services. The reason is the difficulty in building a usable speech user interface (SUI) for a new user group or a challenging acoustic and language situation, without having a good, off-the-shelf speech recognizer available [Laput *et al.*, 2013]: recognizers work reliably when it is known what people say, how they say it, what noises occur, etc. This is hard to achieve for many researchers, developers, and designers who are non-experts in speech recognition, such as application developers or HCI experts. As a result, many studies that HCI researchers have conducted are "Wizard-of-Oz" studies, i.e. using a simulated ASR [e.g., Stifelman *et al.*, 2013; Tewari, 2013]. Unfortunately, while Wizard-of-Oz studies are good for iterating on the design of the application, they cannot uncover real recognition errors and test real system usage [Thomason *et al.*, 2013]. These therefore become the leading source of usability issues in speech applications once the product is released [Lai *et al.*, 2008].

At the same time, research on automatic speech recognition (ASR) has seen great strides in recent times. It has reached a point where given enough speech expertise and in-domain data, a sufficiently accurate speech system can be developed for any scenario, including those of non-native accents, background noise, children's voice, or other similar challenges for speech recognition. However, if a speech recognition system doesn't work as intended, it is generally impossible for a non-expert to tell, for example, if the ASR fails because users speak too slow ("speaking rate" is a frequent cause of mismatch), too "clearly" (i.e. they hyper-articulate), because the background noises are unexpected, or because of some other reason; while an expert can usually analyze the error pattern quickly. Adaptation or several optimizations of an existing recognizer can generally mitigate the problem, and will often result in a functional system [Abras *et al.*, 2004; Preece *et al.*, 2002], but recognizing *what* to do and *how* do it needs tremendous expertise and experience in developing speech recognition systems. These systems also have a large number of "knobs" and parameters, which need to be set correctly in order to achieve satisfactory performance. Application developers usually do not have such expertise or experience and find it difficult to locate people who do, which makes their task even more challenging.

To address the above problem and opportunity, in this thesis, we study the knowledge that automatic speech recognition (ASR) experts use to optimize the accuracy of speech recognizers, and investigate the extent to which it can be made available to non-speech recognition researchers by embedding this knowledge in a toolkit that can generate recommendations for optimizing a speech recognizer. The intended user of our toolkit called Speech Toolkit for Non-Speech Experts (SToNE) is a HCI researcher or a practitioner with a background in computer science (CS), or comparable. He or she is not likely to have the specialized knowledge of how to implement or optimize a speech recognition system, but will be comfortable with data-driven, analytical techniques. We refer to him or her as a non-speech expert or a developer-user in this thesis.

1.1. Current Approaches: Toolkits in Speech Recognition

In the human-computer interaction and speech recognition literature, there are a several tools that have been developed to enable developers to build and test a speech-user interface easily. To enable the reader to quickly understand the nuances, we have summarized this work in Figure 1, and represented them on *expertise required* vs. *functionality achievable* axes.

The first set of tools, e.g. SUEDE [Klemmer *et al.*, 2001] and SPICE [Schultz *et al.*, 2007] require low technical expertise, but are meant to support the design phase and not the development phase. For instance, SUEDE is a toolkit that allows any user-interface designer to

rapidly mock-up a prompt/response speech interface and test it in a Wizard-of-Oz study. It supports an early design phase through the construction and visualization of a dialogue flow, a process that designers earlier did by storyboarding or low-fidelity prototyping. It does not, however, support development of a working recognizer. Another toolkit, SPICE supports development of a baseline recognizer for new languages. It does so by allowing any researcher to input a set of audio files, corresponding phoneme set, and a dictionary to generate the baseline acoustic model. While this is a step towards developing a functional recognizer, it does not support discovery of acoustic or language-specific optimizations, which is key in building a usable recognizer [Van Doremalen *et al.*, 2010; Woodland, 2001; The-imp-of-adaptation, 2014].

On the other end, open-source toolkits such as Kaldi [kaldi], Sphinx [sphinx], and HTK [htk] support development of both the baseline recognizer and an optimized version. These toolkits are at the forefront of speech recognition research, and are widely used across academia and industry by experts in building a speech recognizer. However, any developer who wants to work with these toolkits needs to attain both a formal training in speech recognition as well as tremendous support from experienced users on things they can do to build and iteratively improve the accuracy of their recognizer. As a result, non-expert researchers and developers find them substantially difficult to use. For instance, in 2013 alone, a discussion forum of a widely used recognizer, Sphinx saw over 10,000 posts with over 1000 unique topics from non-experts asking help on various issues of speech recognition, e.g. adapting acoustic models, generating pronunciation variants, etc.

Somewhere in the middle is a set of application programming interfaces (APIs) that are provided by industry groups at Google and Microsoft, among many others. These are relatively simple to use, and don't require formal training in speech recognition. A simple understanding

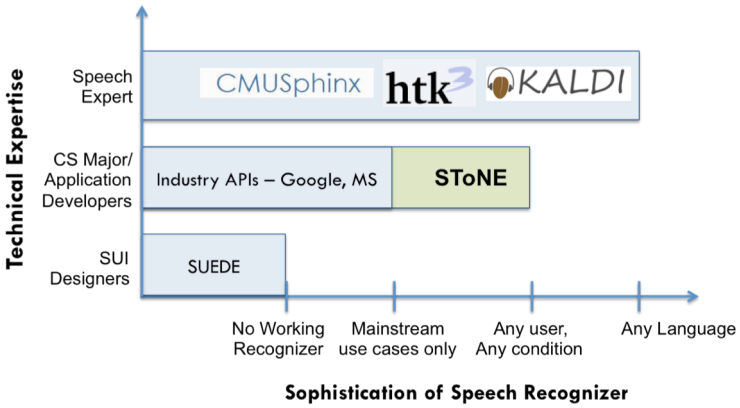


Figure 1.1: A representation of existing speech tools describing the expertise required to use them vs. the functionality that one can achieve.

of API-based development is sufficient. This lends them well to be used by any application developer, e.g. a computer science minor. However, these APIs don't offer much flexibility, and can only work with typical use cases, such as native adult English speakers. Moreover, developers don't have access to the background acoustic and language models installed on the server to perform any adaptations that may be needed for the recognizer to work for their own application's scenario.

A recent publication by human-computer interaction researchers [Laput *et al.*, 2013] illustrates the above issues. Their research group did not have enough expertise to improve the underlying models of existing recognizers, and neither of the existing API's provided them with enough accuracy to start using off-the-shelf. Eventually, their application's usability suffered due to poor speech recognition accuracy for non-native English speakers.

Our toolkit, SToNE targets audience with similar background as users of these APIs, but enables greater functionality, i.e. the ability to develop functional recognizers for any user group who can use it in any acoustic condition. As future work, this can also be extended to any language, but that is out of scope for this thesis. An important note is that SToNE is not a layer on top of industry APIs (as can be misunderstood from Figure 1), but can be viewed as a wrapper on top of the state-of-the-art toolkits, such as Kaldi, Sphinx, or HTK.

1.2. Proposed Approach

Well-trained speech experts have accumulated years of experiential knowledge that guides them "intuitively" in building a speech recognizer for new situations and new users. This knowledge is hard to directly transfer to non-experts or novices. However, we take the view that by observing these experts in-action, we can study and formalize their tacit knowledge. This formalized knowledge can then be used for the benefit of novices for automatic analysis and recommendation of appropriate optimization techniques. Therefore, this thesis aims to solve the above issues by: (i) formalizing expert speech knowledge in form of a rule-based knowledge structure, (ii) providing a set of tools to perform automatic analysis of the degrading context(s) for speech recognition, and explaining regions of high error in the recognizer output, and (iii) providing links to corresponding methods to adapt the recognizer, or even performing some of the adaptation or optimizations automatically. It is important to note that adaptation of speech recognizers does not only consist of picking m suitable adaptations from a set of n techniques ($m <= n$), in the correct order, but may also require changes to the user interface, transcription or additional data collection of a specific type, therefore requiring an understanding of why the recognizer is failing.

1.3. Research Goals

Our overarching research goal is to enable non-speech experts to rapidly develop usable speech applications. Since recognition accuracy is central to usability, and associated optimizations are one of the hardest tasks for non-experts, we focus on studying the knowledge that experts in speech recognition use to optimize the accuracy of speech recognizers, and showing that this knowledge can be made available to non-speech specialists. We identified this opportunity through two research studies in building speech applications, and refer to them as case study #1 and case study #2 below.

- **Case Study #1:** Developed and deployed two speech-recognition-enabled language learning games for rural children in India, and identified challenges associated in developing usable speech-interfaces and opportunities to support similar development by non-speech experts.
- **Case Study #2**: Demonstrated that simple changes to user interaction technique that are made possible after achieving reasonable accuracy can further improve usability of the speech applications. For this, we designed, developed, and user tested a user interaction technique that can improve recognition accuracy for a dictation task over existing dictation styles, while keeping the underlying recognizer constant.
- **Research Goal 1 (RG1)**: Understand and formalize the tacit knowledge that speech experts use to optimize the accuracy of a speech recognizer for acoustic data from various usage contexts. The challenge in studying this knowledge is that it is largely tacit, i.e. experts have internalized it so well that they apply it unconsciously, and hence, find it difficult to articulate this knowledge unless they are actually using it when talking about it. While there are formalisms such as design patterns [Alexander, 1977] for representing expert knowledge, we believe that a toolkit that not only embodies this knowledge, but also helps its user to take advantage of this knowledge is the best way to make it accessible.
- **Research Goal 2 (RG2)**: Understand how, and the extent to which, this knowledge can be made accessible to non-speech experts, by designing a set of tools that incorporate this knowledge: the toolkit will automatically analyze the ASR context, and its user-interface will display visualizations for various system configurations and adaptations based on this analysis. It will allow the non-speech expert to compare his/ her intuition on *who* is using the system (somebody with an accent, children, etc.), *how* it is being used (command and control speech, conversational speech, etc.), and *where* it is being used (mismatch in acoustic/ channel conditions), and suggest appropriate adaptation steps, which the system will then evaluate, once the user has performed them.
- **Research Goal 3 (RG3)**: Understand the extent to which SToNE can assist non-speech experts in improving the accuracy of a speech system, which in turn improves the usability of the application. We will perform summative evaluations where non-experts are asked to

use the toolkit and build a speech recognizer for a challenging use case, i.e. children's conversational speech.

1.4. Thesis Contributions

This work makes contributions to both human-computer interaction and speech recognition technologies as follows:

- Identification of the problem and the opportunity; demonstrate the challenges associated with building a speech recognizer for a non-expert *(Case Study #1, Chapter 2)*: Through two rounds of fields studies, including development and deployment of a speech recognizer for children who were non-native speakers of English, we discuss the challenges involved in building an accurate SUI, and the lessons we can learn to facilitate the development process for other non-expert developers.
- **Formalize the tacit knowledge of speech recognition experts and make it accessible for the benefit of non-speech experts** *(RG1, Chapter 4)***:** We conduct contextual interviews and cognitive task analysis with speech experts to understand the process, knowledge, and intuition that they use while building accurate speech recognizers. We formalize this knowledge in a rule-based knowledge base, and evaluate its predictive power with two English language datasets that have significant challenges for speech recognizer development such as children's speech, background noise, or variable speaking rate. Through the formalization of this knowledge base, we provide a recommendation and analysis engine specific for speech recognizer development.
- **Develop a set of speech analysis and recommendation tools for non-speech experts** *(RG2, Chapter 5)***:** We develop a set of tools that can assist non-speech experts by automatically and semi-automatically analyzing reasons of recognition failure, and then recommending optimization techniques for tackling those issues. These tools include a: (i) feature extraction module for pronunciation scoring, signal-to-noise ratio extraction, speaking rate, etc. that quantifies specific signals for understanding the acoustic context of a given dataset, (ii) an analysis module that identifies the significant degrading factors for the speech recognizer and generates appropriate visualizations for further analysis, and (iii) a recommender module that provides insight into "why the recognizer doesn't work," and then guides non-expert developers on specific steps that can be done to make it work.
- **Demonstrate that non-speech experts can develop accurate and usable speech recognizers with help of SToNE** *(RG3, Chapter 5)***:** We conduct experimental studies that evaluate the extent to which non-speech experts can benefit from the proposed tools for the design and development of accurate and usable SUIs. We expect that the tools developed and proposed in this thesis will lower the entry bar for speech-user interface development considerably.

- Demonstrate how changes in interaction technique – after a sufficiently accurate recognizer is developed – can further improve usability *(Case Study #2, Chapter 3)*: Once a recognizer has been developed with sufficient accuracy, we show that simple edits to the interaction style can lead to additional significant gains in recognition accuracy. To demonstrate this, we design and develop a novel interaction technique called "Voice Typing" that reduces the recognition error rate in modern day dictation interfaces simply by modifying the user's interaction style. This case study demonstrates reasons why speech recognition and HCI communities should be tightly coupled for the development of a usable and useful SUI, and why tools that can make development of accurate speech recognizers simple can be of tremendous value for non-speech experts

1.5. Thesis Organization

This thesis is organized in two parts. The first part (Chapters 2, 3, and 4) describes our background and motivation for this thesis. We discuss two case studies on developing speech recognition applications, and draw lessons for SUI development. In particular, Chapter 2 describes project SMART: Speech-enabled Mobile Assisted Reading Technology, which aimed at developing speech-enabled English language learning games for rural children in India. The speech recognition component of the system here was very challenging (especially for non-experts in speech recognition), and in developing the recognizer for these users we identified the thesis opportunity. Chapter 3 builds on the findings of Chapter 2, and steps further from technical changes to the recognizer to the interaction design changes. Here, we discuss the design and evaluation of a new interaction style "Voice Typing," and show that once a sufficiently accurate recognizer has been developed, it paves way for interaction technique changes, which in turn, further improves overall usability of a speech application. Chapter 4 brings the results from these two case studies together and discusses myths that designers and developers have about SUIs, and provide ground reality for them. This chapter also sets up the problem and solution space, which becomes the basis for thorough investigation in the next part of this thesis.

The second part of this thesis comprises of Chapters 5, 6, and 7. This part describes our efforts in developing **SToNE**: A **S**peech **To**olkit for **N**on-Speech **E**xpert. In Chapter 5, we illustrate results from interviews with speech experts that led to the formalization of a speech recognition accuracy improvement process called ChAOTIC, and an associated rule-based knowledge base. We also discuss results from two lab studies that evaluate this process and the knowledge base on its ability to correctly identify and automatically predict the appropriate optimization techniques. In Chapter 6, we further this work by discussing the overall design and architecture of SToNE. Chapter 7 discussed an in-depth evaluation of SToNE with non-experts and semi-

experts, and summarizes several lessons from the user study, which paves way for future research. Finally, Chapter 8 summarizes the contributions of this thesis, discusses current limitations, and sets forth an agenda for future speech-user interface researchers.

2. CASE STUDY: SPEECH GAMES

This chapter is based on the work presented in:

> Kumar, A., Reddy, P., Tewari, A., Agrawal, R., Kam, M. (2012). Improving literacy in developing countries using speech recognition-supported games on mobile devices. In *Proc. Human Factors in Computing Systems (CHI)*, Austin, USA, ACM, pp. 1149-1158.

In this chapter, we discuss the design, development, and deployment of two English language-learning games for rural children in India. The games use speech recognition on smartphones to recognize user's speech, and give appropriate feedback. The speech recognition component for this application is highly challenging because of multiple reasons, such as non-native accent, children, noisy environment, variable speaking rate of the users, etc. Through the process of developing and deploying these games, we realized the tremendous expertise needed for building a working speech recognizer, and eventually, this case study helped us develop the central research question of this thesis: to enable non-speech experts, such as designers in human-computer interaction, to rapidly develop accurate speech recognizers.

2.1. Need for Speech-enabled Second Language Learning Games

Poor literacy and low educational standards remains a barrier to economic empowerment in many areas of the developing world. Games that teach a standardized curriculum can make a profound impact on the learning needs of such, underserved communities. For instance, two non-government organizations, Pratham and the Azim Premji Foundation [azim-premji-foundation] deployed computer games in the rural areas of India, and demonstrated significant gains on mathematics test scores [Banerjee *et al.*, 2007]. Experiments by the MILLEE project [Kam *et al.*, 2008; Kumar *et al.*, 2010] with rural children in India have also shown promising outcomes with e-learning games for teaching English as a Second Language (ESL). Such games are a perfect instruction delivery source as they provide for an engaging experience while delivering education, therefore ensuring voluntary use by learners [Kumar *et al.*, 2010].

At the same time, a large body of work investigates the utility of speech recognition for developing language learning software. For instance, MIT's literacy tutor [McCandless, 1992]

targets pronunciation skills by providing interactive feedback for poorly articulated or mispronounced words. CMU's project LISTEN [Mills-Tettey *et al.*, 2009] is a reading tutor that listens to children reading, and provides feedback for poor reading. The University of Colorado's Foundation to Literacy Program started a reading program that was designed for beginning and poor readers. It targets foundational reading skills such as letter knowledge, phonological awareness and decoding skills in order to improve listening and reading comprehension. UC Berkeley's SPRING [Tewari *et al.*, 2010] developed a pronunciation tutor on smartphones that helps Hispanic children in the USA to acquire correct English pronunciations. Rosetta Stone [rosetta-stone] and Carnegie Speech [carnegie-speech] use proprietary speech recognition technology to elicit oral responses from learners, and check whether the users have spoken the correct word, phrase, or sentence.

Yet, much of the success of speech-based language learning software has been in the industrialized world. For projects in the developing world, the literacy competencies targeted are far restricted, and don't deal with spoken language assessment. For instance, MILLEE [Kam *et al.*, 2007; Kam, Kumar *et al.*, 2009; Kam, Mathur *et al.*, 2009], Multiple Mice [Pawar *et al.*, 2007], Kane [Dias *et al.*, 2005], and Same Language Subtitling [Arora, 2006] have all demonstrated significant learning gains, but not for spoken language skills. In the developing world, while the focus remains on deploying technologies in real-world scenarios, such as usage in the field, or outside the classroom, the primary deterrent remains the challenge in building a working speech recognizer for acoustic and language conditions, e.g. challenges arising due to dialects, pronunciations, variable speaking rate, deployment in noisy environments, children's voice, etc.

In this chapter, we develop two language learning games with speech recognition support that were deployed in rural India with children from grades 4-5. For our games, we focused on a literacy skill called word reading, which is the ability of the learner to read and understand a written form of the word. We describe 'word reading' next, and illustrate why speech recognition is useful for developing this language learning skill.

2.2. English Language Competency: Word Reading

In our games, we focus on a specific language learning competency, called word reading. Word reading is the ability for the learner to read and comprehend written words, and is fundamental for literacy development. Yet it is one of the most challenging skills to acquire, especially in a second language, because it requires the integration of visual, sound, and meaning information [Perfetti *et al.*, 2001]. In order to address this need, from a second language acquisition perspective, our learning games examine the possible benefits of practicing producing words

aloud, rather than simply reading them receptively in one's mind, for rural Indian students in grades 4-5.

Word reading is a multi-faceted construct that depends on the quality of representation of three linguistic and cognitive sub-systems: orthographic (visual script), phonologic (sound), and semantic (meaning), according to the Lexical Quality Hypothesis (LQH) [Perfetti, 2010]. When a written word is encountered, each orthographic unit must be connected to its appropriate phonological unit, allowing a learner to assign sound information to a word and thus *decode* it accurately. For instance, the letter "c" must be mapped with the sound /k/, "a" with /a/, and "t" with /t/ and so forth. A connection must also be made between this phonological representation (the sound /cat/) and its appropriate meaning (small, furry animal), and thus *semantic extraction* must also occur. If the quality of any of these sub-systems is compromised, then word reading will be hampered, and thus, reading comprehension will be impaired [Perfetti, 2007].

At the same time, it has been argued that oral production is critical for new learners of a language as an oral output provides specific input back to the mind, which in turn assists a learner to transition from declarative knowledge (ability to declare that you know a word) to productive knowledge (ability to fluently use the word) [De Bot, 1996].

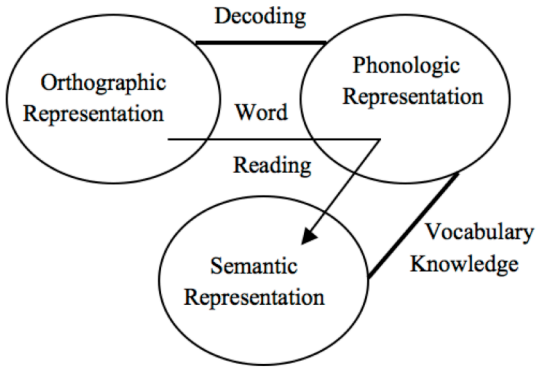


Figure 2.1: The three components of word reading: orthographic, phonologic, and semantic, and two sub-skills comprising word reading: decoding and vocabulary knowledge. Our language learning games test the hypothesis that productive practice (i.e., oral production of vocabulary words while passing the phonologic stage) will lead to better learning gains than just receptive practice. From a technical perspective, supporting productive practice requires the use of speech recognition.

Integrating these strands of research, our conceptual framework is that there are three – orthographic, phonologic, and semantic – representations that are required for word reading, and the connections between them can be processed productively or receptively (see figure 2.1). Based on this framework, we expect that saying the name of a picture out loud (productive

processing) will reinforce semantic extraction more strongly than if the link is receptively processed, i.e. matching a picture and its name [De Bot, 1996; Swain *et al.*, 1995]. Thus, our experiment consisted of two conditions: (1) Receptive (Re), and (2) Productive (Pr). Specifically, we hypothesized that: Productive training will be more beneficial for word reading than Receptive training (H1).

2.3. Game Designs

Building on the growing popularity of educational games [Gee, 2003] and successes with games to teach vocabulary in a second language in the developing world [Kam, Kumar *et al.*, 2009; Pawar *et al.*, 2007], we designed two educational games: Market Game and Farm Game. At the game design level, these games draw inspiration from traditional village games that rural children play on a daily basis and find enjoyable. This was similar to a previous study that had proposed drawing characteristics from local games (compared to those in the contemporary Western videogames), for developing educational games for rural users [Kam, Mathur *et al.*, 2009].

Market and Farm games incorporate actions such as the ability to either *catch* or *evade* a player, both of which were two popular actions in local, physical games. Each learning game followed a teaching, game play, and practice sequence (with or without speech recognition support). This sequence was informed through earlier usability tests with rural children [Kam *et al.*, 2008].

2.3.1. Market Game

In the Market Game, the teaching phase (as shown in Figure 2.2A) entailed introducing the vocabulary words to the user. As in popular commercial software [rosetta-stone], introduction of

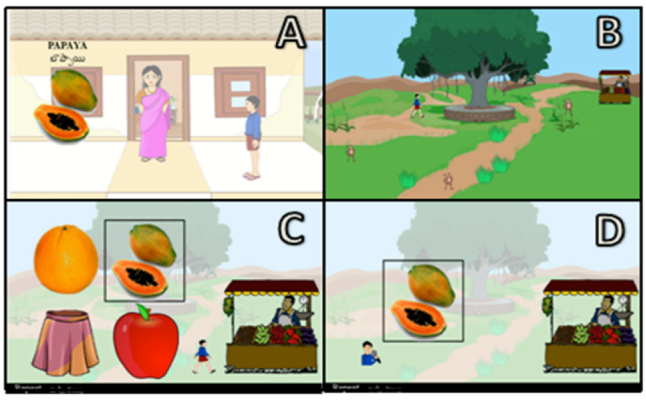


Figure 2.2: These screenshots describe the game sequence for the market game. (A) *Teaching phase*: Introduces the English word for the common nouns, which are the items to be bought in the market. (B) *Game play*: shows the boy attempting to move towards the shop, while the monkeys try to catch him. (C) *Receptive practice condition*: to purchase the item from the shop, the user has to map the word that the software plays aloud with its image from a list of four choices. (D) *Productive practice condition*: to purchase the correct item at the shop, the user has to say aloud the word that corresponds to the image.

a word meant mapping the image of the vocabulary word to its sound, as well as, its native language and English text. At a time, the software introduced five words (one after another), which was based on prior psychological studies that show that an average person can retain 7±2 new items in their short-term working memory at any time [Miller, 1956]. Once the word has been introduced, the user could replay the audio playback for its pronunciation at the teaching phase for any number of times, before going to the next screen.

Next, in the game play screen (Figure 2.2B) of the market game, the aim was to move the boy character from left (home) to right (shop in the market), while avoiding monkeys en-route. This activity was an adaptation of the daily routine of going to the market for children in rural India; and integrated actions from the popular physical games that they play, such as evading an opponent. Next, depending on the experimental condition that the player was assigned to, the player could purchase items from the shop by either *selecting* the correct item that corresponded to the said word (Figure 2.2C, Re condition), or by *saying the word aloud* that corresponded to the image displayed (Figure 2.2D, Pr condition). To recognize user's speech in the conditions that required productive practice (with or without hints), we used pocketSphinx [Huggins-Daines *et al.*, 2006] – an open-source, small footprint, automatic speech recognizer (ASR) for mobile devices. Our choice of pocketSphinx was based on experimental evaluation of mobile speech recognizers [Kumar *et al.*, 2011], where pocketSphinx outperformed other mobile ASRs such as TinySphinx on small vocabulary tasks.

As an alternate, we could run a recognizer in a force-aligned mode to take away the need for implementing a working recognizer, i.e. since we already know what the correct answer is prior to the user speaking it, we could simply identify "how close the spoken answer is to what is expected." The problem of force-alignment is simpler than speech recognition, primarily because you don't have to search through a long list of possible responses to identify a close match. However, force-alignment was unsuitable for our purposes for two reasons: (i) our goal was not only to identify whether the users have spoken the correct word, but also to identify what they have spoken. The latter was needed to understand the types of "learning" errors that they were making, and (ii) in force-alignment, it is difficult to determine an appropriate threshold level that marks whether the answer is acceptable or not. Various distortions like noise, variable speaking rate, children's voice, etc. make this determination even harder.

2.3.2. Farm Game

The Farm Game (Figure 2.3) had the same pedagogical cycle of "teaching, gameplay, and practice", but used a different game action than Market game. The objective of the farm game was to save the farm by "catching" all the thieves and retrieving the items that they had stolen. Again, this activity was based on the common rural India scenario where children helped their

parents keep vigil on the farm in the farming season, where a lot of animals or thieves tend to steal the crop. In order to recover an item from the thief, the user could do it in one of the two ways as described above for the Market Game in Figure 2.2C (receptive practice) or Figure 2.2D (productive practice), depending on the condition that the participant was randomly assigned to.

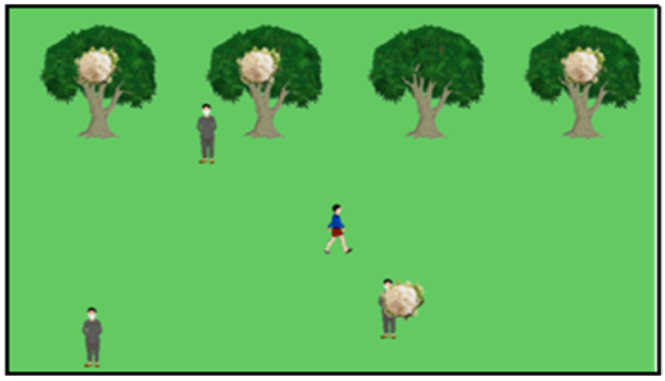


Figure 2.3: This screenshot shows the *game play* screen for the farm game, where the boy character attempts to catch the thief who is stealing the vegetable from the farm.

2.4. Usability Testing

We started off by testing the usability of our system, in terms of both the speech recognition accuracy, and the game designs. 10 children in grades 4-5 in Hyderabad play-tested successive versions of both games. For our initial prototype, we used the standard acoustic and language models provided with pocketSphinx [Huggins-Daines *et al.*, 2006]. These models had been developed from hundreds of hours of audio data by multiple speech recognition experts, and had performed well for adult native English speakers. However, the speech recognition failed miserably in these usability sessions, with word error rates as high as 90% for some speakers. At this point, we decided to recruit a speech expert to guide our efforts in building a better recognizer for our users. This process is described in the next section.

In terms of the speech-interface design for the game, we found that children were confused as to when they should speak. Typically, SUIs indicate this by displaying a microphone icon when it is time to speak. However, from these early usability tests, we realized that traditional icons for prompting the user to speak (A, B, C in Figure 2.4) were unintuitive to participants since most of them were never introduced to the notion of a microphone. Instead of showing the

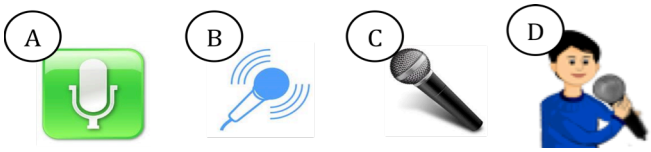


Figure 2.4: Microphone icons tested in the usability tests; A, B, C represent icons that are currently used in existing speech user-interfaces; D represents the icon that was understood by participants of our study.

microphone on its own, they liked the idea of showing a boy holding a microphone, because they were better able to relate to instances of local elections or rallies where people would hold a microphone and speak into it.

2.5. Role of Speech Recognizer Development and Improvements

Due to the multiple challenges of speech recognition for challenging contexts such as ours, off-the-shelf resources were not directly applicable. To account for numerous acoustic variabilities arising due to varying accents, background noise etc. we followed a three-step process to develop the recognizer. This process was guided by the efforts of our speech expert. First, we collected a speech corpus from representative speakers and trained our own baseline acoustic model. Second, we analyzed this corpus for factors that affected the speech recognition accuracy, and third, we adapted the baseline speech recognizer on these factors to improve its accuracy. Below we describe these three steps in more detail.

Our speech corpus was made up of 6250 utterances (~4 hours of speech) from 50 rural Indian children, equally divided across gender and grades 4-5. For training our recognizer, we selected two sets: a set of 25 speakers and a set of 40 speakers. 10 speakers comprising of 5 male speakers was held-out as a testing set for both the above two sets. Each utterance in the dataset was labeled with the word that it represented at the time of recording, and these labeled inputs were used while training the speech recognizer. As in the original pocketSphinx paper [Huggins-Daines et al., 2006], our baseline acoustic model used Hidden Markov Models with a 5-state Bakis topology. They were trained on the 6250 utterances, using 1000 tied Gaussian Mixture Models (senones) and 256 tied Gaussian densities. Since the application required the user to speak words in isolation, we used a unigram statistical language model with a vocabulary size of 25 words.

To improve recognition accuracy beyond the baseline performance of the above acoustic and language models, we next sought to adapt our recognizer to account for the variabilities in users' speech. Since adaptation can happen along several dimensions, we first quantitatively analyzed the speech utterances along seven voice metrics that collectively describe the characteristics of voice, namely articulation (a), speaking rate (r), sound quality (q), pitch (f0), and the first three formant frequencies (f1, f2, f3). Using univariate and multivariate analyses (the description of which is beyond the scope of this paper), the influence of each metric was calculated on the word error rate (WER), which appropriately measures the recognizer's accuracy in isolated word speech recognition tasks, such as ours. Such an analysis was necessary to identify the most critical factors and save time on performing all possible adaptation techniques.

As shown in Figure 2.5, speaking rate and articulation were the most significant factors that impeded recognition performance in the context of our users. Based on this analysis, to account for variations in the speaking rate, we adapted the baseline speech recognizer using a widely used continuous frame rate adaptation technique [Chu *et al.*, 2010]. Similarly, to account for non-native articulations, we added pronunciation variants to the dictionary that were a closer match to the accent of the users than the ones in the baseline dictionary.

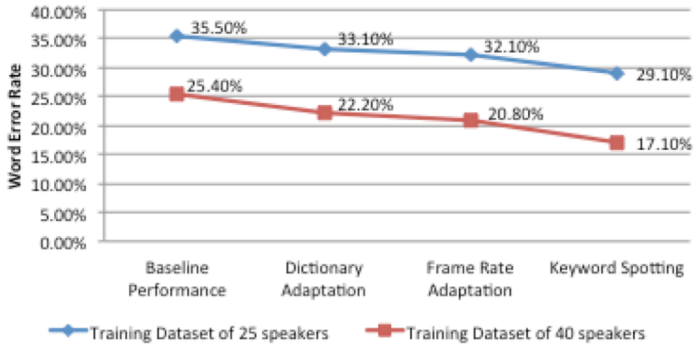


Figure 2.5: Summary of progressive performance improvements when the amount of training data was increased, and with the type of adaptation technique used.

In addition to the above quantitative analysis, we also qualitatively explored the recognition results of the incorrectly recognized words from our usability tests. We observed that in some cases of misrecognized words, the results had extra words – for instance, "papaya" was decoded as "peas papaya cup", perhaps, because background noise at the start and end of actual speech was also decoded. To overcome this, we considered the output of the ASR to be correct if the target vocabulary keyword was spotted anywhere in the decoded sentence. In addition to keyword spotting, we also used noise-canceling microphones to minimize the effect of external noise.

2.6. User Study

Once we had achieved reasonable, we deployed the application to test both the speech recognition accuracy with real users and the learning gains that the application could achieve for word reading. Informed by our theoretical framework as in section 2.2, we had the following hypothesis for this study:

H1: Productive training (Pr) is more beneficial for word reading than receptive training* (Re)

2.6.1. Participants

21 participants (11 boys and 10 girls) took part in our first study in June-July 2010 (5 weeks). They were 9 to 13 years old (mean=10.5 years) in 4th and 5th grades. We selected children in this

age and grade level because word reading skills are increasingly important for reading comprehension and academic success at this level [August et al., 2005]. All participants were attending a public school in a rural area near Hyderabad, India. Telugu and English were taught as their first and second language respectively at this school. Most of their families owned at least one mobile phone.

2.6.2. Experiment Setup

The experiment involved a pre-post test block design. To ensure relevance of the curriculum in the games, each game targeted 10 unique words (all concrete nouns). The 20 words were chosen from government-issued English textbooks for 4th and 5th graders. To avoid introducing a confounding variable from our selection of these words, we consulted teachers at the school to verify that they do not plan to teach these words in their classes at the time that the experiment was planned. Each child played both games, but was randomly assigned to one of the two conditions: Re or Pr. With this assignment, when the 21 participants played the market game, 10 were in the Pr condition and 11 were in Re; and when they played the farm game, 11 were in Pr and 10 were in Re. Each participant played the games in an hour-long, after-school session comprising a word reading pre-test, 15 minutes training session, 30 minutes of gameplay, and a similar word reading post-test immediately after gameplay. Specifically, in the pre- and post-tests, the outcome variable (word reading score) was measured before and after each game using an adapted visual format from the Peabody Picture Vocabulary Test [Dunn *et al.*, 1997], which is used for measuring vocabulary knowledge. In our word reading test, every word was displayed on the screen with four images, and participants were asked to match the word with the picture that best represented its semantics. A correct match adds one to the participant's score, while an incorrect match or the lack of a response does not change the score. In a training session before the study, experimenters explained the games and allowed each participant to play the games with 5 practice words. We will explain how this training session was helpful for participants in the "results for speech recognition" sub-section.

2.6.3. Results for Learning Gains

The means and standard deviations for pre- and post- word reading scores, and post-test gains, are shown in Table 2.1 for both the Re and Pr conditions. As a sanity check, the t-tests did not reveal any significant differences in post-test gains between the market and farm games, for both the Re (p=0.12) and Pr (p=0.48) conditions; we expected both games to exhibit comparable post-test gains since they adopted similar designs and instructional principles. Likewise, between both games, there was no statistically significant difference on pre-test scores (p=0.24) for both conditions, indicating that participants in both conditions started with the same baseline knowledge of word reading. This allowed us to focus on analyzing word reading gains without using pre-test scores as a covariate.

Given the above sanity checks and in order to perform a more robust statistical analysis, for each condition, we combined the scores across both games. On this combined dataset, post-hoc power analysis indicated a power of 0.62 (with cohen's *d* as 0.9 and significance level as 0.05), which was deemed adequate for inferential analysis for our purposes. A one-way 2-factor ANOVA showed a significant difference between the two conditions on post-test gains ($F(1,40) = 5.4$, p=0.001). More specifically, after 30 minutes of game play, gains were observed under both conditions: 1.0 word under Re, against 2.7 words under Pr. These results suggest that with even as little as 30 minutes of game play practicing words – receptively or productively – word reading scores increase significantly. Even more critically, there is evidence that productive training is significantly more beneficial for word reading development than simply receptively practicing words.

Variable	Condition	Market Game	Farm Game
Pre-Test	Re	7.8 (3.5)	6.9 (2.5)
	Pr	8.4 (2.8)	8.9 (3.0)
Post-Test	Re	8.5 (3.7)	8.3 (2.2)
	Pr	10.6 (1.5)	12.1 (3.0)
Gain	Re	.6 (1.1)	1.4 (2.3)
	Pr	2.3 (1.9)	3.2 (1.8)

Table 2.1: Means and standard deviations (in parentheses) for pre-test score, post-test score, and post-test gains for both games, for each of the two conditions (i.e. Re and Pr).

2.6.4. Results for Speech Recognition

After a series of context-based adaptations such as those arising due to non-native accent and noise (as in Figure 2.5), the speech recognizer's accuracy was 17.1%, when the recognizer was trained on data from 40 speakers (Figure 2.5). In other words, approximately, 1 out of 6 times the speech recognizer misunderstood what the user spoke. In practice, we trained the recognizer on the entire dataset of 50 speakers, and noticed that misrecognitions occurred only 1 out of 10 times during real usage. In cases when the user happened to speak the correct answer (but the recognizer misunderstood it to be incorrect), the participants learned to repeat the word from the short, 15-minute training sessions prior to actual gameplay. In general, these training sessions had a dual role: first, to acquaint the participants with the game dynamics using five curriculum words, and second, to help them realize that the system might make recognition mistakes. In order to proceed in the latter case, they could either change their response, or if they were confident, then repeat it. From our observations, most of the times when there was a false negative (i.e. user spoke correctly but speech recognizer misrecognized), learners repeated the

vocabulary word rather than switching their response, and thus system's misrecognition is less likely to have impacted any learning gains.

2.7. Discussion

In this chapter, we have discussed our efforts in developing and deploying a language learning speech recognition application for a challenging user group – children in rural India. At the start of the project when we lacked core speech recognition expertise, we had used an "off-the-shelf" recognizer directly for our application. In this case, our usability tests showed that the application was unusable by the children, as the speech recognizer failed to recognize their speech almost every 9 of 10 times (90% error rate). Thereafter, with the help of analysis and speech optimizations guided by a speech expert, we were able to improve this error rate to 17.1%. Moreover, we demonstrated learning gains for the users with as much as just 30 minutes of game play, which was the primary purpose of the application. The findings in this chapter suggest two learnings: (i) off-the-shelf recognizer may not work directly, and improvements to the underlying models are often needed that are dependent on the user group, and (ii) it is possible to improve speech recognition accuracy by several orders of magnitude, provided there's an expert understanding of what to do, and a small collection of audio files from representative users. Our experiments have also demonstrated that with the help of regression analysis, we can identify specific issues that impact the speech recognition most, and then using expert help, we can perform specific optimizations to improve recognition accuracy.

In the next chapter, we shift focus from optimizing recognizer models to interaction technique edits for the benefit of usability. We show that once we have developed a reasonable recognizer, interaction technique edits become possible, and those edits in turn lead to further improvements in the usability and recognition accuracy of the speech system. In Chapter 4, we bring the lessons of these two case studies together to identify and provide more understanding on SUI myths, as perceived by interaction and usability designers.

3. CASE STUDY: VOICE TYPING

This chapter is based on the work presented in:

> Kumar, A., Paek, T., Lee, B. Voice Typing: A new speech interaction model for dictation on touchscreen devices. (2012). In *Proc. Human Factors in Computing Systems* (CHI), Austin, USA, ACM, pp. 2277-2286.

This chapter shifts focus from developing recognizers (as in Chapter 2) to designing interaction techniques for achieving recognition accuracy gains. In this chapter, we show that once a reasonable accuracy has been attained through the necessary optimizations to the recognizer, changes to the interaction technique are made possible, and those edits lead to additional significant accuracy improvements. In other words, initial accuracy gains, achieved through optimizations, enable interaction technique edits that weren't possible earlier. Conversely, interaction technique edits enables accuracy gains that aren't possible through optimizations beyond an accuracy level.

To demonstrate the above, we focus on improving the interaction style of current large vocabulary continuous speech recognition (LVCSR) dictation systems such as those from Nuance, IBM, AT&T, Apple, Vlingo, etc. The current interaction model for dictation tasks follows a *voice recorder* metaphor where users must first formulate what they want to say in utterances, and then produce them, as they would with a voice recorder. These utterances do not get transcribed until users have finished speaking (as indicated by a pause or via push-to-talk button), at which point the entire output appears at once after a few seconds of delay. This interaction model is adopted because at the surface, it provides technical benefits: the recognizer can have as much context as possible to improve decoding. In other words, real-time presentation of output is sacrificed for accuracy. Users must then break their train of thought, verify the output verbatim and correct errors. This process can be mentally disruptive, time-consuming, and frustrating. Indeed, users typically spend only 25-30% of their time actually dictating. The rest of the time is spent on identifying and editing transcription errors [Karat *et al.*, 1999; MacKenzie *et al.*, 2002].

We introduce Voice Typing, a new speech interaction model where users' utterances are transcribed as they produce them to enable real-time error identification. For fast correction, users leverage a gesture-based marking menu that provides multiple ways of editing text. Voice Typing allows users to influence the decoding by facilitating immediate correction of the real-time output. The metaphor for Voice Typing is that of a secretary typing for you as you monitor and quickly edit the text using the touchscreen. The changes needed to enable the above interaction style are not complex, but have a high impact on the accuracy gains, as well as the usability. For example, as you will read below, the only parameter we changed in the state-of-the-art dictation interfaces was the "detection of silence" from 1-2 seconds (default) to 0 seconds.

3.1. Related Work: Real-time Speech Decoding & Error Correction Techniques

A wide variety of input methods have been developed to expedite text entry on touchscreen devices. Some of these methods are similar to speech recognition in that they utilize a noisy channel framework for decoding the original input signal. Besides the obvious example of handwriting recognition and prediction of complex script for languages such as Chinese, a soft keyboard can dynamically adjust the target regions of its keys based on decoding the intended touch point [Gunawardana *et al.*, 2010]. The language model utilized by speech recognition to estimate the likelihood of a word given its previous words appears in almost all predictive text entry methods, from T9 [Grover *et al.*, 1998] to shape writing techniques [Zhai *et al.*, 2003] such as SWYPE [swype].

Beyond touchscreen input methods that are similar to speech recognition, a few researchers have explored how to obtain more accurate recognition hypotheses from the word lattice so that they can be presented in real-time. Fink *et al.* (1998) found that providing more right context (i.e., more acoustic information) could improve accuracy. Likewise, Baumann *et al.* (2009) showed that increasing the language model weight of words in the lattice could improve accuracy. Selfridge *et al.* (2011) took both of these ideas further and proposed an algorithm that looked for paths in the lattice that either terminated in an end-of-sentence (as deemed by the language model), or converged to a single node. This improved the stability of hypotheses by 33% and increased accuracy by 21%. Note that we have not yet tried to incorporate any of these findings, but consider this part of our future work.

With respect to the user experience of obtaining real-time recognition results, Aist *et al.* (2007) presented users with pre-recorded messages and recognition results that appeared either all at once or in an incremental fashion. Users overwhelmingly preferred the latter. Skantze *et al.* (2009) conducted a similar study where users recited a list of numbers. Again, users preferred to review the numbers in an incremental fashion. All of this prior research justifies the Voice

Typing speech interaction model. To our knowledge, the user study we describe in the Section 3.3 represents the first attempt to compare incremental, real-time transcription with traditional dictation on a spontaneous language generation task using LVCSR decoding.

The Voice Typing gesture-based marking menu is related to research in multimodal correction of speech recognition errors. In Martin *et al.* (1980), preliminary recognition results were stored temporarily in a buffer, which users could interactively edit, by spoken dialogue or by mouse. Users could delete single words or the whole buffer, re-speak the utterance, or select words from an n-best list. Suhm *et al.* (2001) proposed switching to pen-based interaction for certain types of corrections. Besides advocating spelling in lieu of re-speaking, they created a set of pen gestures such as crossing-out words to delete them. Finally, commercially available dictation products for touchscreen devices, such as the iPhone Dragon Dictation application, also support simple touch-based editing. To date, none of these products utilize a marking menu.

3.2. Voice Typing

The speech interaction model of Voice Typing follows the metaphor of a secretary typing for you while you monitor and correct the text. Real-time monitoring is important because it modulates the speed at which users produce utterances. Just in the way that you would not continue speaking if the secretary were lagging behind on your utterances, users of Voice Typing naturally adjust their speaking rate to reflect the speed and accuracy of the recognizer. Indeed, in an exploratory design study we conducted where participants dictated through a "noisy microphone" to a confederate (i.e., an experimenter), we found that as the confederate reduced typing speed (to intentionally imply uncertainty about what was heard), participants also slowed their speaking rate. For the prototype we developed, the transcription speed was such that users produced utterances in chunks of 2-4 words (see the Prototype section for more details). In other words, for real-time feedback, in Voice Typing, users are not restricted to speaking one word-at-a-time with a brief pause in between, as in discrete recognition [Rosen, 1997], nor required to wait for long after speaking full utterances, like in traditional dictation. Instead, users can speak in small chunks that match their thought process. Ideally, if the recognizer could transcribe as quickly as users could produce speech with perfect accuracy, the Voice Typing experience would be more like dictating to a professional stenographer. However, with current state-of-the-art recognition where corrections are required due to misrecognitions, Voice Typing is more akin to dictating to a secretary or a fast-typing friend.

We now elucidate the motivation for Voice Typing and the technical challenges required to realize its full potential. We also describe the prototype we implemented for touchscreen devices.

3.1.1. Technical and Cognitive Motivations

Voice Typing is motivated by both cognitive and technical considerations. From a cognitive standpoint, human-computer interaction researchers have long known that providing real-time feedback for user actions not only facilitates learning of the user interface but also leads to greater satisfaction [Payne, 2009]. With respect to natural language, psycholinguists have noted that as speakers in a conversation communicate, listeners frequently provide real-time feedback of understanding in the form of back-channels, such as head nods and "uh-huh" [Clarke *et al.*, 1991]. Indeed, research suggests that language processing is incremental i.e. it usually proceeds one word at a time, and not one utterance or sentence at a time [Altmann *et al.*, 1999; Tanenhaus *et al.*, 1995; Traxler *et al.*, 1997], as evidenced by eye movements during comprehension. Real-time feedback for text generation is also consistent with the way most users type on a keyboard. Once users become accustomed with the keyboard layout, they typically monitor their words and correct mistakes in real-time. In this way, the interaction model for Voice Typing is already quite familiar to users.

From a technical standpoint, Voice Typing is motivated by the observation that dictation errors frequently stem from incorrect segmentations (though to date we are unaware of any published breakdown of errors). Consider the classic example of speech recognition failure: "It's hard to wreck a nice beach" for the utterance "It's hard to recognize speech." In this example, the recognizer has incorrectly segmented "recognize" for "wreck a nice" due to attaching the phoneme /s/ to "nice" instead of "speech." Because having to monitor and correct text while speaking generally induces people to speak in small chunks of words, users are more likely to pause where segmentations should occur. In other words, in the example above, users are more likely to utter "It's hard <pause> to recognize <pause> speech," which provides the recognizer with useful segmentation information. In the Experiment section, we assess whether Voice Typing actually results in fewer corrections.

Voice Typing has yet another potential technical advantage. Since there is an assumption that users are monitoring and correcting mistakes as they go along, it is possible to treat previously reviewed text as both language model context for subsequent recognitions and supervised training data for acoustic [Gales, 1998] and language model adaptation [Bellegarda, 2004]. With respect to the former, real-time correction prevents errors from propagating to subsequent recognitions. With respect to the latter, Voice Typing enables online adaptation with acoustic and language data that has been manually labeled by the user. There is no better training data for personalization than that supervised by the end user.

3.1.2. Prototype

While the technical advantages of Voice Typing are appealing, implementing a large vocabulary continuous speech recognition (LVCSR) system that is designed from scratch for Voice Typing is no small feat and will likely take years to fully realize (see the Related Work section). At a high level, decoding for LVCSR systems typically proceeds as follows (see [Rabiner et al., 1993] for a review). As soon as the recognizer detects human speech it processes the incoming audio into acoustic signal features, which are then mapped to likely sound units, e.g. phonemes. These sound units are further mapped to likely words and the recognizer connects these words together into a large lattice or graph. Finally, when the recognizer detects that the utterance has ended, it finds the optimal path through the lattice using a dynamic programming algorithm or Viterbi [Rabiner, 1989]. The optimal path yields the most likely sequence of words (i.e., the recognition result). In short, current LVCSR systems do not return a recognition result until the utterance has finished. If users are encouraged to produce utterances that constitute full sentences, they will have to wait until the recognizer has detected the end of an utterance before receiving the transcribed text all at once. This is of course the speech interaction model for traditional dictation.

In order to support the speech interaction model of Voice Typing, the recognizer would have to return the optimal path through the lattice created thus so far. This can be done through *recognition hypotheses*, which most speech APIs expose. Unfortunately, for reasons that go beyond the scope of this paper, recognition hypotheses tend to be of poor quality. Indeed, in building a prototype, we explored leveraging recognition hypotheses but abandoned the idea due to low accuracy. Instead, we decided to use LVCSR decoding as is, but with one modification. Part of the way in which the recognizer detects the end of an utterance is by looking for silence of a particular length. Typically, this is defaulted to 1-2 seconds. We changed this parameter to 0 milliseconds. The effect was that whenever users paused for just a second, the recognizer would immediately return a recognition result. Note that the second of delay is due to other processing the recognizer performs.

To further facilitate the experience of real-time transcription, we coupled this modification with two interaction design choices. First, instead of displaying the recognition result all at once, we decided to display each word one by one, left to right, as if a secretary had just typed the text. Second, knowing the speed at which the recognizer could return results and keep up with user utterances, we trained users to speak in chunks of 2-4 words.

Providing real-time transcriptions so that users can monitor and identify errors is only the first aspect of Voice Typing. The second is correcting the errors in a fast and efficient manner on touchscreen devices. To achieve this goal, we leveraged a marking menu that provides multiple

ways of editing text. Marking menus allow users to specify a menu choice in two ways, either by invoking a radial menu, or by making a straight mark in the direction of the desired menu item [Kurtenbach *et al.*, 1994]. In Voice Typing, users invoke the marking menu by touching the word they desire to edit. Once they learn what choices are available on the marking menu, users can simply gesture in the direction of the desired choice. In this way, marking menus enables both the selection and editing of the desired word, and provides a path for novice users to become expert users. Figure 3.1(a) displays the marking menu we developed for Voice Typing. If users pick the bottom option, as shown in Figure 3.1(b) they receive a list of alternate word candidates for the selected word, which is often called an *n-best list* in the speech community. The list also contains an option for the selected word with the first letter capitalized. If they pick the left option, they can delete the word. If they pick the top option, as shown in Figure 3.1(c) they can re-speak the word or spell it letter by letter. Note that with this option they can also speak multiple words. Finally, if they pick the right option, as shown in Figure 3.1(d) they can add punctuation to the selected word. We decided to include this option because many users find it cumbersome and unnatural to speak punctuation words like "comma" and "period." Having a separate punctuation option frees users from having to think about formatting while they are gathering their thoughts into utterances.

It is important to note that Voice Typing could easily leverage the mouse or keyboard for correction, not just gestures on a touchscreen. For this chapter, however, we decided to focus on marking menus for touchscreen devices for two reasons. First, a growing number of applications on touchscreen devices now offer dictation (e.g. Apple's Siri which uses Nuance Dragon [nuance], Android Speech-to-Text [Steele *et al.*, 2010], Windows Phone SMS dictation [windows-7-speech-recognition], Vlingo Virtual Assistant [vlingo], etc.). Second, touchscreen devices provide a unique opportunity to utilize touch-based gestures for immediate user feedback, which is critical for the speech interaction model of Voice Typing. In the user study below, our comparison of marking menus to regular menus reflects the correction method employed by almost all of these new dictation applications.

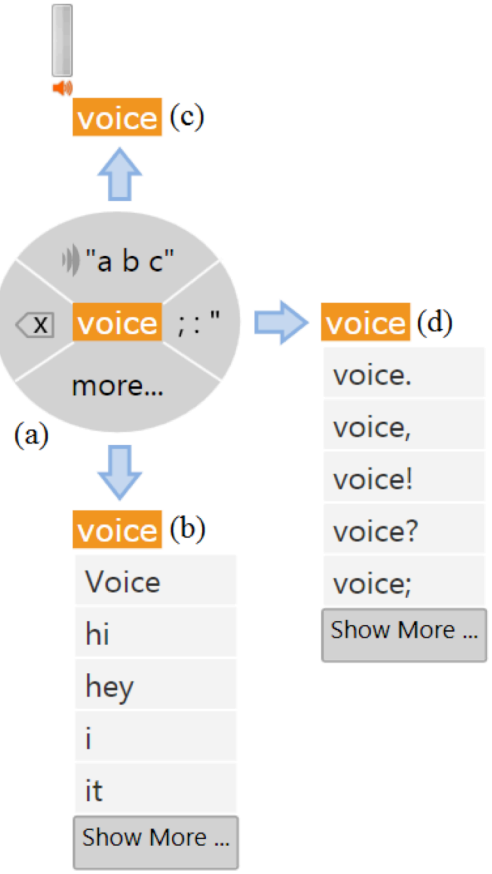


Figure 3.1: Screenshots of (a) the Voice Typing marking menu, (b) list of alternate candidates for a selected word, including the word with capitalized first letter, (c) re-speak mode with volume indicator, and (d) list of punctuation choices

3.3. User Study

In order to assess the correction efficacy and usability of Voice Typing in comparison to traditional dictation, we conducted a controlled experiment in which participants engaged in an email composition task. For the email content, participants were provided with a structure they could fill out themselves. For example, "Write an email to your friend Michelle recommending a restaurant you like. Suggest a plate she should order and why she will like it." Because dictation entails spontaneous language generation, we chose this task to reflect how end users might actually use Voice Typing.

3.3.1. Experimental Design

We conducted a 2x2 within-subjects factorial design experiment with two independent variables: *Speech Interaction Model* (Dictation vs. Voice Typing) and *Error Correction Method* (Marking Menu vs. Regular). In Regular Error Correction, all of the Marking Menu options were made available to participants as follows. If users tapped a word, the interface would

display an n-best list of word alternates. If they performed press-and-hold on the word, that invoked the re-speak or spelling option. For deleting words, we provided "Backspace" and "Delete" buttons at the bottom of the text area. Placing the cursor between words, users could delete the word to the left using "Backspace" and the word to the right using "Delete." Users could also insert text anywhere the cursor was located by performing press-and-hold on an empty area.

The order of presentation of *Speech Interaction Model* and *Error Correction Method* was counter-balanced. We collected both quantitative and qualitative measures. With respect to quantitative measures, we measured rate of correction and the types of corrections made. With respect to the qualitative measures, we utilized the *NASA task load index (NASA-TLX)* [Hart, 2006] because it is widely used to estimate perceived workload assessment. It is divided into six different questions: mental demand, physical demand, temporal demand, performance, effort, and frustration. For our experiment, we used the software version of NASA-TLX, which contains 20 divisions, each division corresponding to 5 task load points. Responses were measured on a continuous 100-point scale. We also collected qualitative judgments via a post-experiment questionnaire that asked participants to rank order each of the four experimental conditions (Dictation Marking Menu, Voice Typing Marking Menu, Dictation Regular and Voice Typing Regular) in terms of preference. The rank order questions were similar to NASA-TLX so that we could accurately capture all the dimensions of the workload assessment. Finally, we collected open-ended comments to better understand participants' preference judgments.

3.3.2. Software and Hardware

We developed the Voice Typing and Dictation *Speech Interaction Models* using the Windows 7 LVCSR dictation engine. As mentioned before, for Voice Typing, we modified the silence parameter for end segmentation via the Microsoft System.Speech managed API. In order to control speech accuracy across the four experimental conditions, we turned off the (default) MLLR acoustic adaptation. Both types of *Error Correction Methods* were implemented using the Windows 7 Touch API and Windows Presentation Foundation (WPF). We conducted the experiment on a HP EliteBook 2740p Multi-Touch Tablet with dual core 2.67 GHz i7 processor and 4 GB of RAM.

3.3.3. Participants

We recruited 24 participants (12 males and 12 females), all of whom were native English speakers. Participants came from a wide variety of occupational backgrounds (e.g. finance, car mechanics, student, housewife, etc.). None of the participants used dictation via speech recognition on a regular basis. The age of the participants ranged from 20 to 50 years old (M = 35.13) with roughly equal numbers of participants in each decade.

3.3.4. Procedure

In total, each experimental session lasted 2 hours, which included training the LVCSR recognizer, composing two practice and three experimental emails per experimental condition, and filling out NASA-TLX and post-experiment questionnaires. To train the LVCSR recognizer, at the start of the each session, participants enrolled in the Windows 7 Speech Recognition Training Wizard, which performs MLLR acoustic adaptation [Gales, 1998] on 20 sentences, about 10 minutes of speaking time. We did this because in our early usability tests, we found that without training, recognition results were so inaccurate that users became frustrated regardless *Speech Interaction Model* and *Error Correction Method*.

During the training phase for each of the four experimental conditions, the experimenter walked through the interaction and error correction style using two practice emails. In the first practice email, the experimenter demonstrated how the different *Speech Interaction Models* worked, and then performed the various editing options available for the appropriate *Error Correction Method* (i.e., re-speak, spelling, alternates, delete, insert, etc.). Using these options, if the participant was unable to correct an error even after three retries, they were asked to mark it as incorrect. Once participants felt comfortable with the user interface, they practiced composing a second email on their own with the experimenter's supervision. Thereafter, the training phase was over and users composed 3 more emails. At the end of each experimental condition, participants filled out the NASA-TLX questionnaire. At the end of the experiment, they filled out the rank order questionnaire and wrote open-ended comments.

3.4. Results

3.4.1. Quantitative

In order to compare the accuracy of Voice Typing to Dictation, we computed a metric called *User Correction Error Rate* (UCER), modeled after Word Error Rate (WER), a widely used metric in the speech research community. In WER, the recognized word sequence is compared to the actual spoken word sequence using Levenshtein's distance [Levenshtein, 1966], which computes the minimal number of string edit operations–substitution (S), insertion (I), and deletion (D)–necessary to convert one string to another. Thereafter, WER is computed as: WER = $(S + I + D) / N$, where N is the total number of words in the true, spoken word sequence. For a more thorough discussion on the WER, please refer to Section 5.3.

In our case, measuring WER was not possible for two reasons. First, we did not have the true transcript of the word sequence – that is, we did not know what the user had actually intended to compose. Second, users often improvised after seeing the output and adjusted their utterance formulation, presumably because the recognized text still captured their intent. Moreover, we

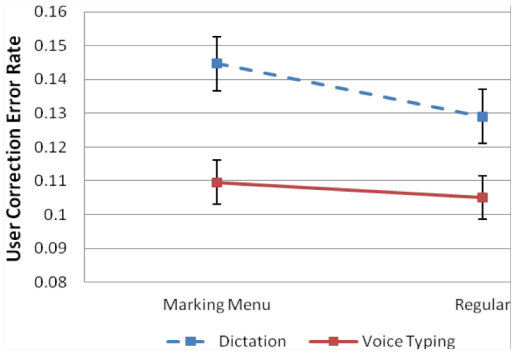


Figure 3.2. User correction error rate for all four conditions. Blue data points are for traditional Dictation, red data points for Voice Typing.

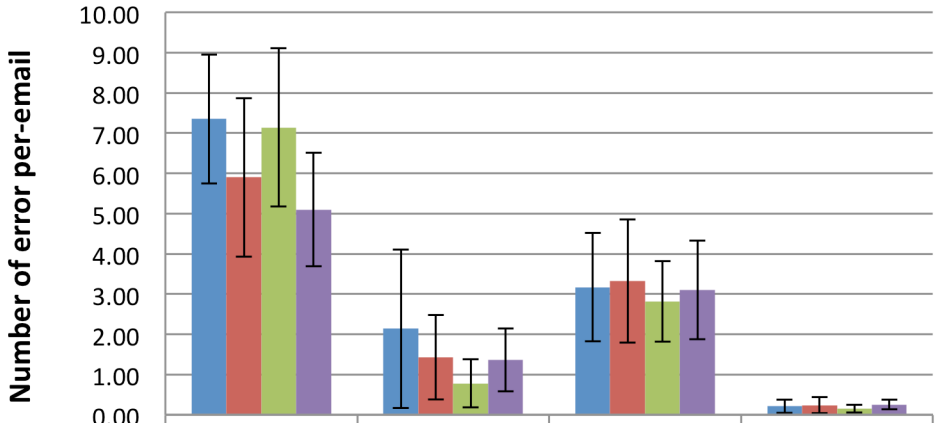
believe that although WER accurately captures the percentage of mistakes that the recognizer has made, it does not tell us much about the amount of effort that users expended to correct the recognition output, at least to a point where the text was acceptable. The latter, we believe, is an important metric for acceptance of any dictation user interface. Thus, we computed UCER as:

$$UCER = \frac{Sub + Ins + Del + Uncorrected}{Num}$$

where *Sub* is the number of substitutions the user made using the 'Respeak' mode (both spelling and re-speaking the entire word) or via using the alternates, *Ins* is the number of word insertions, *Del* is the number of word deletions, *Uncorrected* is the number of words that user identified as incorrect but did not correct due to difficulties in error correction (see the Procedure Section), and *Num* is the number of words in the final text that was submitted by the user.

While the UCER metric captures the amount of effort users expended to correct mistakes, it does not include the errors that were left "unidentified and uncorrected" by the user. For example, there were occasions when words were recognized as plural, instead of singular, but were left unchanged. This may be because they were unidentified or because the user did not feel that the grammatically incorrect text affected the intended meaning. In any case, these cases were rare and more importantly, they were equally distributed across the experimental conditions.

In terms of UCER, a repeated measures ANOVA yielded a significant main effect for the *Speech Interaction Model* ($F(1,46) = 4.15$, $p < 0.05$), where Voice Typing (M = 0.10, SD = 0.01) was significantly lower than Dictation (M = 0.14, SD = 0.01). Looking at Figure 3.2, it may seem as if Marking Menu had slightly higher UCER than Regular, but we did not find any main effect for *Error Correction Method*, nor did we find an interaction effect.



	Substitutions	Insertions	Deletions	Uncorrected
Dictation, Marking Menu	7.35	2.14	3.17	0.21
Dictation, Regular	5.90	1.43	3.32	0.24
Voice Typing, Marking Menu	7.14	0.78	2.82	0.15
Voice Typing, Regular	5.10	1.36	3.10	0.25

Figure 3.3. Average number of substitutions, insertions, deletions made by the user in order to correct an email for each of the four experimental conditions, and the number of words left uncorrected.

In terms of types of corrections, as described previously, UCER consists of four different types of corrections. We wanted to further tease apart the types of corrections that were significantly different across experimental conditions to understand what led to lower UCER in Voice Typing than Dictation. Figure 3.3 plots average errors per-email for all four conditions. For substitutions, We obtained a significant main effect for *Error Correction Method* ($F(1,46) = 5.9$, $p < 0.05$), where Marking Menu had significantly higher substitutions (M = 7.24, SD = 0.5) than Regular (M = 5.50, SD = 0.5). For insertions, deletions, and identified but uncorrected errors, we did not find any significant effects.

To contrast the amount of time users had to wait to see the recognition result, although this was not a dependent variable, we measured the delay from the time the recognizer detected speech (Time (speech)) to when the actual transcription (Time (text)) was shown in the user interface:

$$Delay = \frac{\sum_{all\ emails} Time\ (text) - Time\ (speech)}{Total\ Number\ of\ Emails}$$

As shown in Figure 3.4, the average waiting time for Voice Typing was 1.27 seconds, which of course was significantly lower than that of Dictation, 12.41 seconds. Although the delay in dictation seems ten times large, we should note that this includes the time that the user took to speak the entire utterance, as well as the delay time.

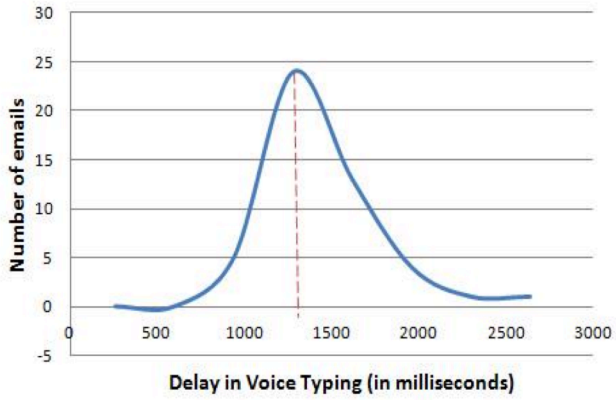


Figure 3.4. Frequency distribution of the system response times across emails in Voice Typing condition. For most emails the delays were within one standard deviation (0.3 seconds) of the average (1.27 seconds), and all the emails were with two standard deviations from the average.

For Voice Typing, we were interested to see if the average delay varied across emails either due to the users' speaking style or other acoustic differences that might have led to a difference in user experience. Surprisingly, all the delays were within two standard deviations from the average, with 37 (out of 48 emails for Voice Typing) having delays within one standard deviation from the average i.e., between 1.27 ± 0.3 seconds, as shown in Figure 3.4.

3.4.2. Qualitative

3.4.2.1. Voice Typing vs. Dictation

A repeated measure ANOVA on the NASA TLX data for mental demand yielded a significant main effect for *Speech Interaction Model* ($F(1, 46) = 4.47$, $p = 0.03$), where Voice Typing (M = 30.90, SD = 19.16) had lower mental demand than Dictation (M = 39.48, SD = 20.85). Furthermore, we found a significant main effect *Speech Interaction Model* on effort and frustration ($F(1,46) = 4.03$, $p = 0.04$ and $F(1,46) = 4.02$, $p = .05$, respectively). In both cases, Voice Typing displayed significantly lower effort and frustration than Dictation.

On the rank order questionnaire, overall 18 out of 24 participants indicated a preference for Voice Typing over Dictation ($\chi^2(1) = 7.42$, $p < 0.01$). Furthermore, 18 participants ranked Voice Typing as having less effort than Dictation ($\chi^2(1) = 3.99$, $p < 0.05$), and 17 ranked Voice Typing as having less frustration ($\chi^2(1) = 9.53$, $p < 0.05$). As indicated, all of the above rankings were statistically significant by Wilcoxon tests.

Open-ended comments highlighted the motivation for the speech interaction model of Voice Typing. As one participant put it, *"It [Voice Typing] was better because you did not have to worry about finding mistakes later on. You could see the interaction [output] as you say;*

thereby reassuring you that it was working fine." On the other hand, not all participants agreed. One participant explained: *"I preferred Dictation, because in Voice Typing, if one word was off as I was speaking, it would distract me."*

3.4.2.2. Marking Menu vs. Regular Menu

On the NASA-TLX data, we found a significant main effect for *Error Correction Method* on physical demand ($F(1,46) = 4.43$, $p = 0.03$), where Marking Menu (M = 19.50, SD = 20.56) had significant lower physical demand than Regular Menu (M = 28.13, SD = 19.44).

On the rank order questionnaires, 21 out of 24 participants overall preferred Marking Menu to Regular Menu ($\chi^2(1) = 25.86$, $p < 0.01$). Furthermore, 21 participants ranked Marking Menu as having less mental demand than Regular Menu ($\chi^2(1) = 22.3$, $p < 0.01$) and 21 ranked Marking Menu as having less physical demand ($\chi^2(1) = 22.3$, $p < 0.01$). As indicated, all of the above rankings were statistically significant by Wilcoxon tests.

With respect to open-ended comments, participants seemed to appreciate the menu discovery aspect of marking menus. As one participant put it, *"It [Marking Menu] was great for a beginner. It was easier mentally to see the circle with choices and not have to concern myself with where to select my [error correction] choices from."* Another participant highlighted how Marking Menu allowed both selection and correction at the same time: "*It [Marking Menu] seemed to involve less action.*" Again, not everyone agreed with 3 participants claiming that they found Marking Menu to be challenging to learn. In order to understand why these 3 participants felt that way, we went through their video recordings. It turned out that these participants had larger fingers than most, and had difficulties selecting and swiping (up, down, or left) words, particularly single letter words like "a."

3.5. Discussion

In this chapter, we introduced Voice Typing, a new speech interaction model where users' utterances are transcribed as they produce them to enable real-time error identification. Our experimental results indicated that Voice Typing had significantly fewer corrections than Dictation even though the acoustic and language models for the recognizer were the same for both *Speech Interaction Models*. A plausible explanation is that when users correct transcriptions in real-time, this prevents errors from propagating. Users also felt that Voice Typing exhibited less mental demand, perceived effort, and frustration than Dictation.

With respect to *Error Correction Methods*, while there was no significant difference in the overall user correction error rate between the two methods, we found that users used substitutions (re-speak, spell, or alternates) significantly more in Marking Menu than Regular,

the current style of error correction in the dictation interfaces today. One plausible explanation for this is that since users found gestures to be less mentally and physically demanding, they preferred to substitute the word rather than leave it uncorrected. This is evidenced by the trend that people left more words uncorrected in the Regular Menu method than Marking Menu. Another plausible explanation is that the current interfaces required edit operations at a word level even if successive words were incorrectly recognized. This could have potentially led to more substitutions than needed. We plan to explore phrase-level edit operations as future work.

Despite the above positive findings for Voice Typing, we do not believe that the Dictation interaction model should be completely dismissed. There is merit in using a *voice recorder* style of interaction when the context demands a "hands-free, eyes-free" interaction, such as driving. Surely, in these cases, Voice Typing would not be a feasible approach. Also, in other cases such as highly populated public spots, we imagine that typing on a soft keyboard might still be preferred for privacy.

3.6. Conclusion

In this Chapter, we discussed a novel interaction technique called "Voice Typing," that displays real-time results of user's conversation speech. This real-time display of recognizer's results enabled real-time error correction, which stopped further error propagation. On the high level, we showed that once a reasonable accuracy has been attained through the necessary optimizations to the recognizer, changes to the interaction technique are made possible, and those edits lead to additional significant accuracy improvements. In other words, initial accuracy gains, achieved through optimizations, enable interaction technique edits that weren't possible earlier. Conversely, interaction technique edits enables accuracy gains that aren't possible through optimizations beyond an accuracy level. In the next Chapter, we bring together the results from Chapters 2 and 3 to discuss and debunk myths that SUI developers, designers, and researchers have about speech application development. In doing so, we set a research agenda, which paves way for the next part of this thesis.

4. SUIs: MYTHS *vs.* REALITY

In this chapter, we synthesize our experience from the previous two Chapters. We identify four myths that, according to our own experiences, anecdotal evidences from a panel on Speech and HCI at ACM CHI 2013 [Munteanu *et al.*, 2013], and a workshop at ACM CHI [Munteanu *et al.*, 2014] SUI developers, designers, and researchers have about SUIs and their development. We then provide ground reality in terms of an "inside the black box" explanation or debunk the myth using results from the above two case studies.

Myth #1: No data like more data

US government-sponsored projects such as GALE [Olive *et al.*, 2010] are strong promoters of the philosophy that the best way to improve speech recognition accuracy is by collecting large amounts of data. Google and Apple also follow the same philosophy. While it is true that collection of more training data will improve recognition accuracy, it is highly expensive, and sets an extremely high bar for new research groups and under-represented parts of the population to participate and contribute in the speech recognition development process.

Fortunately, collection of additional training data is not the sole way of improving recognition accuracy. In fact, recent research efforts such as IARPA's Babel [babel] programs attempt to shift the focus towards developing a functional recognizer with the minimum amount of resources required. In case study #1, we have shown that carefully selected optimizations to the underlying acoustic and language models, with the help of experts, can lead to large gains in accuracy. At the same time, while the optimizations to the recognizer models may be required, they alone can be insufficient. In case study #2, we show that once a reasonable accuracy has been attained through the necessary optimizations, changes in the interaction technique are possible, and those changes can lead to additional significant accuracy improvements. This is important because initial accuracy gains, achieved through optimizations, enable interaction technique changes that weren't possible earlier. Conversely, interaction technique change enables accuracy gains that aren't possible through optimizations beyond an accuracy level.

Collecting additional training data, however, is still useful. Our recommendation is that an initial usable application must be developed (through the necessary optimizations and interaction technique edits), and over-time, data can be collected as users use the speech-user interface. In certain systems, this data collection can also be governed by the "human in the loop" philosophy, i.e. enabling the users to correct the system if the recognition results are

incorrect. This gives user-transcribed data for retraining the system. In case study #2, we implemented a marking menu to enable rapid, real-time error correction for dictation tasks, which not only stopped error propagation in speech decoding, but also gave us user-labeled training data for improving our speech recognizer further.

Yet, optimizations to the underlying recognizer, identifying possible interaction edits, and retraining the system can be challenging, especially for developers who have no speech recognition expertise. A toolkit that facilitates and guides non-expert developers, designers, or researchers in this task can certainly be beneficial.

Myth #2: SUIs are unusable by non-native speakers, children, or in noisy environments

Recent papers in human-computer interaction have pointed that SUIs become unusable for users such as non-native speakers of English [Laput *et al.*, '13] primarily because the users "expect" for the recognizer to fail. In fact, the paper reported that:

> *"Non-native English participants often had a preconceived belief that their accent would impede their performance with the speech interface. Thus, they tended to use the speech interface less frequently [than native English speakers]. Moreover, they also reported being more self-conscious about their pronunciations, and the presence of a native English speaker (the user-study facilitator in this case) could have limited their inclination to use the speech interface."* [Laput *et al.*, '13]

Often, such preconceived notions deter interaction designers from integrating speech modality in their applications. The reality is that there's nothing intrinsically unrecognizable about any user's speech or any environment. What matters most is the mismatch between the training data and the actual testing conditions. Off-the-shelf models bundled with popular open-source recognizers such as Sphinx [sphinx] or Kaldi [kaldi] are often only trained on native, adult English accent in quiet environments, and are as such unusable by any other type of user or in any other environment. Similarly, APIs offered by Google [google-speech-api] or Microsoft [ms-sapi] are not directly usable for non-native English speakers or in noisy environments.

Yet, we can achieve reasonable accuracy for most users, languages, or environments, at least to a point where misrecognitions don't deter application use. In case study #1, we have shown that by collecting an initial audio dataset from representative users, and by performing necessary optimizations to the recognizer, it is possible to achieve reasonable accuracy, at least to a point where recognizer's ability to misrecognize words does not affect application's usability. In case study #2, we have shown that interaction edits can help restructure the type of input that the

recognizer receives, which in turn, can further boost the recognition accuracy and usability.

However, a speech expert is often needed to understand the reasons of failure, and make the required optimizations. Such optimizations are dependent on the peculiar characteristics of the users who use the system or the environment in which it is used. In case study #1, we have shown that with the help of existing machine learning techniques, we can quantify and identify major reasons of error. We focus our next efforts on developing tools that can incorporate such techniques, and enable non-speech experts to rapidly develop accurate speech recognizers for any user or environment.

Myth #3: SUIs need to achieve very high accuracy before they are usable

Although somewhat related to myth #2, this myth focuses more generally on the question "*what is the acceptable accuracy for a SUI to be usable?*" Seen in the light of traditional GUIs and the "what-you-see-is-what-you-get" paradigm that are hardly error-prone, the expectations for SUIs are very high. A recent publication at CHI, however, pointed out that:

> *"…achieving perfectly accurate speech processing is a lofty goal that is often nothing short of a fairy tale*" [Munteanu *et al.*, 2013]

> *"…there is now sufficient evidence that many real-life applications using speech technologies* ***do not require 100% accuracy to be useful****… Engaging the CHI community now is timely – many recent commercial applications, especially in the mobile space, are already tapping the increased interest in and need for natural user interfaces (NUIs)* ***by enabling speech interaction in their products****."* [Munteanu *et al.*, 2014].

In fact, there's prior research evidence pointing to the fact that proper interaction design can complement speech processing in ways that compensate for its less-than-perfect accuracy [Oviatt, 2003; Munteanu *et al.*, 2006], or that in many tasks where users interact with spoken information, where word-to-word transcription of speech is not required [Penn *et al.*, 2008].

In our case studies, we have reaffirmed the above findings. In case study #1, the system misrecognized every 1 word (out of 10 words), yet it did not seem to deter learning gains from the application. In cases of false negatives, i.e. when the learner-user spoke the correct answer but the recognizer misunderstood it to be incorrect, the participants had two options: they could either change their response, or if they were confident, then repeat it. From our observations, most of the times when there was a false negative, the participants repeated the vocabulary word rather than switching their response. The participants seemed to have identified this issue in the

initial training sessions, and thus system's misrecognition did not seem to have an impact on the usefulness of the system, although the usability can be improved further by improving recognition accuracy. In case study #2, we have shown that we can use interaction techniques edits as a tool to improve recognition accuracy even further, and subsequently improve usability.

The fundamental question still remains, i.e. "*how good is good enough for speech recognition?*" The answer, in our opinion, is that it depends on the application task and user group. For instance, a study conducted by Munteanu *et al.* (2006) showed that transcripts with up to 25% word error rate are acceptable as useful for end-users in a transcription task for lectures. Of course, here "usefulness" is compared in comparison to alternatives such as manual transcription (which can be expensive), or no transcription (which can mean loss of notes). In other tasks where there are other sources of information, often an even higher WER is acceptable. For instance, in a spoken dialog system, a 52% WER system was usable as other sources of information such as dialog structure was used to achieve good task success rates [Lopes et al., 2013]. Often a moderately high WER system (~25-40% WER) is used to field a "semi-working" system to collect more and better data, which in turn, can be used to further improve accuracy.

Moreover, human-human speech conversations are error prone as well. Eventually, what matters most is setting the correct user expectation. If the users expect "what-you-**say**-is-what-you-get" paradigm, then SUIs have the potential to fail miserably. However, as we showed in case study #1, initial training can help set correct expectations. Often just repeating the words, or switching to another modality, i.e. spelling mode, or keyboard can help reduce user frustration, and eventually improve usability of the system.

Myth #4: Audio Data Collection for SUIs is prohibitively expensive[1]

In order to develop accurate SUIs, audio data collection from representative users is unavoidable. This is especially true for user groups for whom the research community does not already have established datasets. However, at the surface, this task can be prohibitively expensive as proper audio data collection is a research challenge in and of itself. This task, in turn, consists of two (somewhat independent) steps: (i) speech acquisition, i.e. recording audio files from representative speakers, and (ii) transcription, i.e. obtaining text that corresponds to the speech in the audio files. The challenge in the first step is to setup up a correct recording environment, e.g. correct bit-rate, file format, etc., and the challenge in the second step is to transcribe all voices, including non-speech sounds (hmm…, umm…, <breath>), and half-words

[1] Expense can be either in terms of a direct monetary expense or in terms of time value for the project personnel.

that were not either completed by the user, or were merely truncated because of a recording error. Often, one or more of these challenges make the larger task of audio data collection expensive for researchers.

To address this, speech researchers have recently experimented with utilizing the "wisdom of crowds" (or crowdsourcing) on online platforms such as Amazon Mechanical Turk (AMT) as a cheap way to collect and transcribe audio files [Callison-Burch *et al.*, 2010; Freitas *et al.*, 2010; Paek *et al.*, 2007; Parent *et al.*, 2011]. The advantages are many. First, researchers can reach participants that are not co-located with them, e.g. a researcher in US can setup a task for a participant in India. Second, the recording environments are not controlled for the type of microphone or the distance of the microphone from the mouth, which represents the real conditions of speech recognition use. And more importantly, such platforms only require micropayments, e.g. $0.005 per-audio file, as the task is perceived to be a compensation for participants' spare time and not working hours. This, in fact, is vastly cheaper than traditional lab-based recording environments where participants often receive payments in excess of $100/hr.

However, the there are two primary challenges with crowdsourcing-based audio data collection: cognitive load in task construction, and quality control of the responses received. For task construction, researchers need to be careful in setting cognitively feasible tasks. A task may be simple, such as writing down the word that was heard or repeating what was heard. Or it may be more complex such as writing down the sentence that was pronounced with non-linguistic sounds such as coughing being annotated at the same time as the linguistic content. When creating a task, it is recommended that researchers should analyze whether it involves some complex set of decisions that can be divided into separate simpler tasks [Parent *et al.*, 2011]. For instance, in the above example, there should be two passes to annotate the sentence with one pass for the linguistic content and another for the non-linguistic sounds.

The issue of quality control is of equal importance. Quality check can be done at many stages, such as before the task (via a pre-test), during the task (via setting up gold standards for a few responses and checking the participant's response against this standard), or after the task (via another crowd-sourced validation task). While either of these would seem an obvious choice, surprisingly, a survey of 29 publications in 2011 [Parent *et al.*, 2011] revealed that 38% of the researchers did not perform any type of quality control, which led to construction of a poor quality (and somewhat unusable) corpus for them. Conversely, the authors who followed one of the above quality control measures demonstrated that the quality of their audio corpus and transcription was almost as good as that collected by experts [Parent *et al.*, 2011].

Yet, not all user groups and conditions can be covered by crowdsourcing platforms. For instance, in the example of users in case study #1, rural children will not be available to on online crowdsourcing platforms such as AMT or any other, which renders data collection for such user groups to be even more challenging. In our efforts in case study #1, we hired an expert to make regular field trips to the record audio files from representative speakers. While this method can still be expensive, it is more monetarily feasible than scheduling participants to come to a lab for recording sessions. Moreover, once a working prototype has been setup, additional data collection can occur during its deployment and usage by real users.

5. FORMALIZING EXPERT KNOWLEDGE

This chapter is an extension of the work presented in:

Kumar, A., Metze, F., Wang, W., Kam, M. (2013). Formalizing expert knowledge for developing accurate speech recognizers. In *Proc. InterSpeech*, Lyon, France, ISCA, pp. 1121-1125.

The focus of this chapter is to understand and formalize the tacit knowledge that speech experts use to optimize the accuracy of a speech recognizer (corresponds to Research Goal 1). Once formalized, we evaluate the knowledge base on its ability to predict correct optimization steps for both seen and unseen datasets. Accordingly, we present two sub-contributions in this chapter: first, we detail the process and results from interviews with professional speech developers that led to formalizing expert intuition and know-how in a rule-based knowledge base. Second, using two datasets, we outline empirical results that demonstrate the ability of the knowledge base in predicting successful optimization steps to a non-expert. Finally, we discuss how this knowledge base connects with the set of tools (Research Goal 2) that we develop next.

5.1. Methodology: Contextual Interviews with Speech Experts

As is common in knowledge engineering literature [Schreiber *et al.*, 2000] for eliciting knowledge from experts, we conducted semi-structured interviews with five experts, all senior PhD students at CMU. Later in this chapter, we call them "old" experts. The interview methodology draws on ideas from distributed cognition, which posits that knowledge lies not only with the individual, but is also distributed over the tools in the individual's social and physical environment. Thus, our interviews covered questions on experts' interaction with: (i) other individuals, e.g. data collectors, application designers, or other experts, and (ii) with the machines, tools, data sources, etc., e.g. listening to audio files using an audio processing tool. We divided the interviews in two phases: first, we asked the participants to describe a general adaptation process, the common challenges they faced, the people they consulted, and the tools they used. For the second phase, we observed them in-action for a speech recognition optimization task. We gave each expert a dataset that contained utterances from Indian children, recorded in a noisy background on a mobile phone. In section 5.4, we refer to this dataset as Dataset A. Since this dataset contained several potential degrading factors, e.g. noise, children's

speech, accent, etc., we felt it was a good choice to understand an expert's thought process while developing recognizers for challenging contexts. We asked each expert to explain the steps (similar to retrospective think aloud) that they would take to build the best recognizer on this dataset.

Post interviews, the transcripts of the above interviews became the basis for line-by-line open coding process to identify relevant concepts and themes that enhanced our understanding of the optimization process and the associated intuition [Strauss *et al.*, 2008]. Specific instructions or intuitions by the experts were formulated as rules. Once formulated, the same experts vetted these rules for consistency and accuracy of formulation.

5.2. Related Work: Expert Systems and Rule-based Knowledge Representation

A wide variety of expert systems that model experts' knowledge for a specific domain have been successfully developed in the past, e.g. see review by [Liao, 2005] and by [Compton, 2013]. Depending on how the knowledge is represented in such system, they are classified in one of the following four categories: decision trees, rules, semantic nets, or schemata. Of these, decision trees and rule-based systems are typically preferred when the entire knowledge space is difficult to exhaustively define at the start, and there's a requirement to add additional rules or criterion as and when new knowledge becomes available. On the contrary, semantic nets and schemata systems perform well if the entire knowledge structure is well defined at the beginning, and there's a need to capture semantics, e.g. capturing inheritance relationships between entities.

From our interviews with speech experts, we realized that our knowledge representation should meet the following requirements:

- Support iterative, incremental development because new techniques might get developed after the knowledge base is formed,
- Should be functional with incomplete knowledge at the beginning, at least for the acoustic and language situations it has seen before,
- Support conflicting recommendations from the experts,
- Be able to learn from its mistakes, and
- Reflect the structure of knowledge that's tacitly followed by human experts.

Given the above requirements, our knowledge representation follows a rule-based structure, primarily for three reasons: (1) Ease of representation: we recognized that experts had developed many "unspoken rules" that followed an "IF … THEN … ELSE …" structure, (2)

Scalability: previous research in rule-based systems has shown that it is relatively simple for knowledge engineers (or domain experts) to add new techniques in a rule system [Richards, 2009]. This would be of importance, given the ever-expanding needs of speech recognition, and (3) Rule based systems are easier to maintain and modify than decision trees for complex knowledge domains, such as the one we are pursuing.

At the same time, rule-based expert systems have been tremendously popular in many important domains. For instance, in the medical domain, MYCIN [Shortliffe, 1976] was one of the first expert systems that recommended antibiotics for bacterial infection. MYCIN took a set of symptoms as input, found the best rule that matched the criteria, and then, recommended the corresponding antibiotic. In a study conducted, MYCIN out-performed the recommendations made by several doctors [Shortliffe, 1976]. Since then, we have seen a surge in rule-based expert systems. Morgan Stanley announced an increase of $1 million in profit after the installation a rule-based expert system to assist in security trading and portfolio analysis. Many others were developed for recommending mortgages [Steinmann *et al.*, 1990], planning personal finance [Dirks *et al.*, 1995], forecasting climate [Rodionov *et al.*, 1999], plant process control [Acosta *et al.*, 2001], tax consultation [Shpilberg *et al.*, 2012], legal consultation [Clancy *et al.*, 1989], etc. In the computational domains, expert systems have been developed for recommending correct cleansing technique in big data [Dani *et al.*, 2010], for finding issues in code that limit interoperability of web interfaces [O'Connor *et al.*, 2005], to assist in chip design [Bourbakis *et al.*, 2002], to guide development of planar robots [Sen *et al.*, 2004], and in tutoring systems [Hatzilygeroudis *et al.*, 2004]. To our knowledge, no expert system to date has been developed for guiding the development of an accurate speech recognition system.

5.3. Primer to Speech Recognition

ASR is generally thought of as a problem of pattern classification. Namely, we are attempting to classify an observation (or utterance) O as belonging to a class W, where W is a set of words best representing the observation [Rabiner *et al.*, 1993]. The words in W are drawn from a finite vocabulary, but the number of combinations possible is tremendously large. In other words, we want to find the word sequence W that has the maximum probability given the observation O, i.e. we want to maximize $P(W|O)$. This task is often referred to as large-vocabulary, continuous speech recognition (LVCSR). By using Bayes rule, this search space can be restated as:

$$W' = \arg max_W P(W|O) = \arg max_W \frac{P(O|W)P(W)}{P(O)}$$

Since $P(O)$ is constant and given, the above equation simplifies to finding the best hypothesis W' such that:

$$W' = \arg max_W P(O|W)P(W)$$

By restating the fundamental problem of speech recognition in this way, we have restated the problem into components that is easier to model. $P(O|W)$ is referred to as the acoustic model, and $P(W)$ is the language model. These two models form the fundamental components of a speech recognizer. In addition, a pronunciation dictionary forms a third important component that maps each word in the vocabulary to a phoneme sequence, i.e. pronunciation of the word, and is used in the development of the acoustic model. The quality of these two models along with that of the pronunciation dictionary determines the overall quality of the speech recognizer.

The acoustic model is created by taking a set of audio files with corresponding transcripts as input, and using speech recognition toolkits such as Kaldi [kaldi] or Sphinx [sphinx] to create a statistical representation of the sounds that make each word. Sometimes these representations are at the word level, which means that the smallest unit is a word in the acoustic model; however such models become rare for any recognizer having a vocabulary size of greater than 100 words or so. Typically the representations are at the phoneme level, which means that each word has to also have a word-to-phoneme mapping – this is the pronunciation dictionary. In this way, scaling up to thousands of words is possible, as the phoneme set is usually small in a language, e.g. English has a phoneme set of 42 phones.

The language model is created by taking in a corpus of text comprising of naturally occurring sentence structures, e.g. transcription of broadcast news, etc. It is developed using software such as SRILM [Stolcke, 2002] or LMSharp [lmsharp] It assigns a probability to a sequence of m words by means of probability distribution. For instance,

$$P(W) = P(w1\ w2\ w3 \ldots wm)$$

There are many types of language models: in general, a n-gram model accounts for n-1 words history. For instance, a unigram model will compute:

$$P(W) = P(w1)\ P(w2)\ P(w3) \ldots P(wm)$$

Similarly, a bigram model will compute:

$$P(W) = P(w1)\ P(w2\ |w1)\ P(w3|w2) \ldots P(wm|wm-1)$$

Although a high value n in n-gram is preferred as it accurately captures how the language is structured, it also requires more text to achieve reliable probability distributions. Typically

LVCSR systems use trigram models, and apply a number of optimization techniques to further improve reliability.

Many aspects make developing accurate LVCSR difficult. Some reasons are:

- There are no clear boundaries between words in the input set of observations *O*.
- The pronunciation of words can vary considerably due to the context, user, or both. This affects the quality of the pronunciation dictionary and the acoustic models.
- Since the search space of *W* is extremely large in LVCSR, it is not possible to search the entire space for near real-time recognition, and heuristics are used to determine a smaller search space. It's possible that the correct hypothesis lies outside this reduced search space.
- Factors such as noise, speaking rate, hyper articulation, accented speech, children's voice, etc. can tremendously impact speech recognition as the signal is often distorted or mixed with other sources of information, e.g. noise.

A mismatch between training and test set is often one of the leading causes of a poor speech recognizer. The purpose of this thesis is to automatically identify reasons of error occurring in speech recognition results (which could be from either acoustic modeling issues, language modeling issues, pronunciation dictionary issues, or all), and use those to assist non-speech experts in developing accurate speech recognizers.

5.3.1. Word Error Rate

How do you assess quality of a speech recognizer? In the speech recognition community, there is a widely used metric known as Word Error Rate, which measures how error prone the system is. To calculate WER, the recognized word sequence is compared to the actual spoken word sequence using Levenshtein's distance [Levenshtein, 1966], which computes the minimal number of string edit operations–substitution (S), insertion (I), and deletion (D) necessary to convert one string to another. Thereafter, WER is computed as: WER = $(S + I + D) / N$, where N is the total number of words in the true, spoken word sequence.

In terms of the typical values for WER, it really depends on the task. If the task is that of continuous speech recognition with a very large vocabulary (>100,000 words), then 10% WER is excellent, although systems exhibiting WERs upto 30% are also usable [Munteanu *et al.*, 2006]. If on the other hand, it is an isolated word recognition task for a tiny vocabulary size (<100 words), even a 10% WER system is considered not so good. Additionally, for some systems, the WER may exceed 100% where the total number of errors exceeds the total number of words actually spoken. Such systems are rare.

5.4. Knowledge Formalization

Problems in speech recognizer development are varied and numerous, including collecting appropriate data, performing several sanity checks and cleaning operations on the training and test data, properly understanding the usage situation, i.e. the users and the usage environment, and then iteratively performing correct optimizations. Fortunately, there are many "signals" that expert researchers look at when picking the correct technique from the gamut of methods available, which we detail below. Moreover, from the analysis and coding of important themes from the interviews, we realized that most techniques that expert researchers followed could be classified into specific categories, and each category of techniques was followed in a specific process. We captured and formalized this information in a "process framework" that we call ChAOTIC, short for Check data, Adapt, Optimize, Tune, and Integrity check. To sum, the analysis and amalgamation of the above interviews led to two important results: (A) a step-by-step process framework that experts follow while optimizing for accuracy, and (B) a set of rules that guide their choices at each step in this process. Below we detail each of them.

5.4.1. ChAOTIC Process Framework and Rules

A typical accuracy optimization process starts with experts receiving a small set of audio data files (and associated transcripts) recorded in the representative setting of the application use. In the absence of a "universal recognizer", these files are necessary for two main reasons: first, speech experts are typically not data collectors or field researchers i.e. they don't interact directly with end-users of speech applications and may not necessarily have access to the users that they develop recognizers for. Sometimes, the users may be thousands of miles away, e.g. a researcher based in USA developing a recognizer for rural users in India. Second, and more important, these audio files are necessary to train and adapt existing high-quality recognizers (or in some cases, train a completely new recognizer), which is specific for the application's usage situation. The expert's task, then, is to understand and analyze the application situation, and develop a recognizer with the best possible accuracy using the data available. Of course, more data can be collected, which might improve accuracy further, but in most circumstances, a working recognizer is needed to facilitate further data collection by deploying a working prototype.

Here, we describe the Checkdata-Adapt-Optimize-Tune-Integritycheck components that comprise the ChAOTIC process framework. This sequence also reflects the same steps that a speech recognition expert (or a team) usually takes when building an accurate recognizer for new set of users and acoustic or language situations. Note that while each expert might have their own methodology, the below method reflects a typical process, as amalgamated from the interviews with speech experts, and also our own experiences as described in Chapters 2 and 3. Because of the high interdependency between each step, this process is also iterative, i.e. for

instance, experts can return back to step 2 “Adapt”, after they have finished running a specific step in step 3 “Optimize”.

Step 0: Baseline

This step involves selecting an appropriate starting point for the development process. It could be an off-the-shelf recognizer that matches the characteristics of the required recognizer, or a baseline recognizer developed from the training dataset. This step is critical because properly selecting a baseline can become vital to the type of performance that the recognizer finally achieves.

The expert’s decision is usually dependent on what the application’s intended purpose of use is (command-and-control vs. dialogue system), how many speakers it is built for (speaker-dependent vs. speaker independent), how much data is available (1hr, 5hrs, 10hrs, or 50hrs), and what the vocabulary list size is (tiny: 60 words, or small: 500 words). For each such decision point, we documented how the experts handled the specific situation, e.g. in the above case for using an existing recognizer vs. training a new acoustic model, the rule documented was:

> “*IF no matching ‘off-the-shelf’ baseline model found, AND*
> *(app. = command-and-control for single speaker, AND tune data > 1 hour of recording, AND vocabulary size < 60 words), OR*
> *(app. = command-and-control system for many speakers, AND tune data > 5 hours of recording, AND vocabulary size < 60 words), OR*
> *(app. = dialogue system for single speaker, AND tune data > 10 hours of recording, AND vocabulary size < 500 words), OR*
> *(app. = dialogue system for many speakers, AND tune data > 50 hours of recording, AND vocabulary size < 500 words)*
>
> *THEN train your own acoustic model,*
> *ELSE use an existing ‘off-the-shelf’ acoustic model”*

Step 1: Check and clean data

Data obtained from field researchers is usually the following: a set of audio files, corresponding transcripts, a list of words that they want recognized in the final application (called a vocabulary list), and a file containing metadata of the users, e.g. demographics, device information where each recording was made, etc. From the perspective of the speech recognition expert, this data, in its raw form, can have many problems. For instance, transcripts can have spelling errors or fragmented words, such as “alphabe..” or “alphab..” for the word “alphabet”. For fragmented words, it’s difficult to ascertain whether these are merely spelling errors, or whether they

represent an actual half-word in the corresponding audio. In the former case, the transcript needs to be edited, and in the latter case, the fragmented word needs to be added to the list of vocabulary words. Another pre-processing step that needs to be done is to check if all the audio files are recorded using the same frequency and same bit-rate. Depending on what the final goal is for the application, the necessary audio files would have to be down-sampled (more common) or up-sampled (rare).

Other rules take into consideration other characteristics of the decision making process, and an entire list of the characteristics that the knowledge base covers is given below in "Characteristics of Knowledge Base".

Step 2: Adaptation or Training

Once the initial data and corresponding files have been checked and cleaned for any potential errors, the experts usually proceed with one of two steps:

- Training a new acoustic and language model: for the acoustic model, this involves appropriately splitting the data into a train and a validation set, and applying standard "pipeline" of techniques, such as: feature extraction, context-independent and dependent models, speaker adaptive training, discriminative training, etc. While these standard pipelines may differ from tool to tool, but one exists for nearly every toolkit, which is constantly updated to incorporate the latest techniques. Language modeling training is far more standardized. All the experts we interviewed followed the more-or-less the same process of collecting all the training transcripts in one single file, and using a widely accepted standardized tool, SRI Language Modeling Toolkit [Stolcke, 2002] to obtain an initial language model.
- Adapting an existing "off-the-shelf" acoustic and language model: the advantage of starting with an existing high-quality "off-the-shelf" recognizer is that experts already have a functional recognizer as a baseline, and this model needs to be adapted to better understand the peculiar characteristics of the users or the environment represented in the data from the field researchers. The critical point to note here is that the techniques involved in adaptation perform "general" linear transformations of the underlying model parameters (e.g. mean and variance of HMMs in the existing model) closer to the characteristics of the data from the field researchers. They do not, however, apply specific optimization techniques necessary to deal with specific issues, such as accent or noise, which is dealt in step 3. Several rules were documented for this step. E.g.:

  "*IF Adaptation Data Size < 20 sec per-speaker THEN perform Global mean-only MLLR"*
  "*IF Adaptation Data Size > 100 utterances per-speaker, THEN first perform MLLR using first 100 files, and next switch to MAP*."

In addition, to picking the correct transformation technique, this step also accounts for anomalies or missing data files from field researchers. For instance, if the transcripts were missing for the audio files provided (in this case, the adaptation data), this rule would fire:

"IF Transcripts = none, THEN perform two-pass adaptation"

Two-pass adaptation works like this: it performs two runs. In the first run, the existing recognizer is used to decode the adaptation audio files and arrive at a hypothesis transcript for each audio file. In addition to the hypothesis, the recognizer outputs a confidence score in its result. These results (hypothesis and confidence numbers) are used in a second pass to adapt and transform the parameters of the original model closer to the adaptation dataset.

Step 3: Optimize, dependent on the usage context and the user

Adaptations (as in step 2) are necessary but insufficient. Moreover, the general transformations above apply only to the acoustic and language model, and leave the pronunciations dictionary unaltered, when in fact, all are equally important. In contrast to "general" transformations in step 2, this step concerns applying optimization and normalization techniques that relate to specific reasons of failure or degrading factors, e.g. accent, noise, speaking rate, etc. In doing so, the objective is to first identify reasons of where the recognizer is failing, and then apply techniques most suitable to tackling them. Unfortunately, experts we interviewed did not have a quantitative way of identifying such degrading factors, and applied guesstimates by manually listening to a few audio files that were incorrectly recognized and mentally comparing them to those used in the baseline model to understand the point(s) of differences. This trend was also found in a number of recent papers we reviewed [Van Doremalen *et al.*, 2011; Tewari, 2013; etc.].

In our experiments (as we will discuss in the next section), we automate the above qualitative process. The goal is to see whether an automated method could identify similar issues as the manual inspection by experts. To do so, we extract quantitative values for several degrading factors for each audio file in the test set, and its recognition result from the recognizer after step 2. Next, we perform univariate and multivariate regression to identify the impact of each factor on recognition accuracy [Haberman, 1979]. Based on the statistical significance ($p<0.05$), we identify the significant factors, and based on the coefficient of correlation, we rank them from most to least impact. Next, we perform corresponding adaptations for the significant degrading factors, as enlisted in the rules from the expert interviews. E.g.

"*If degrading factor = SNR, perform Cepstral Variance Normalization"*

"If degrading factor = F0, perform Vocal Tract Length Normalization", etc.

Step 4: Tune recognizer parameters

This step includes tweaking recognizer parameters such as insertion penalty, silence penalty, or changing the language model weight based on inspecting the type of errors in the recognition output from step 3. It also includes analyzing confusion pairs, and possibly recommending alternative phrases to change the interface dialogue for reducing common errors.

Step 5: Integrity Check

This step includes checking if the models have been trained on sufficient data, e.g. each Gaussian in the speech recognizer acoustic model should be trained or tuned on a minimum amount of data.

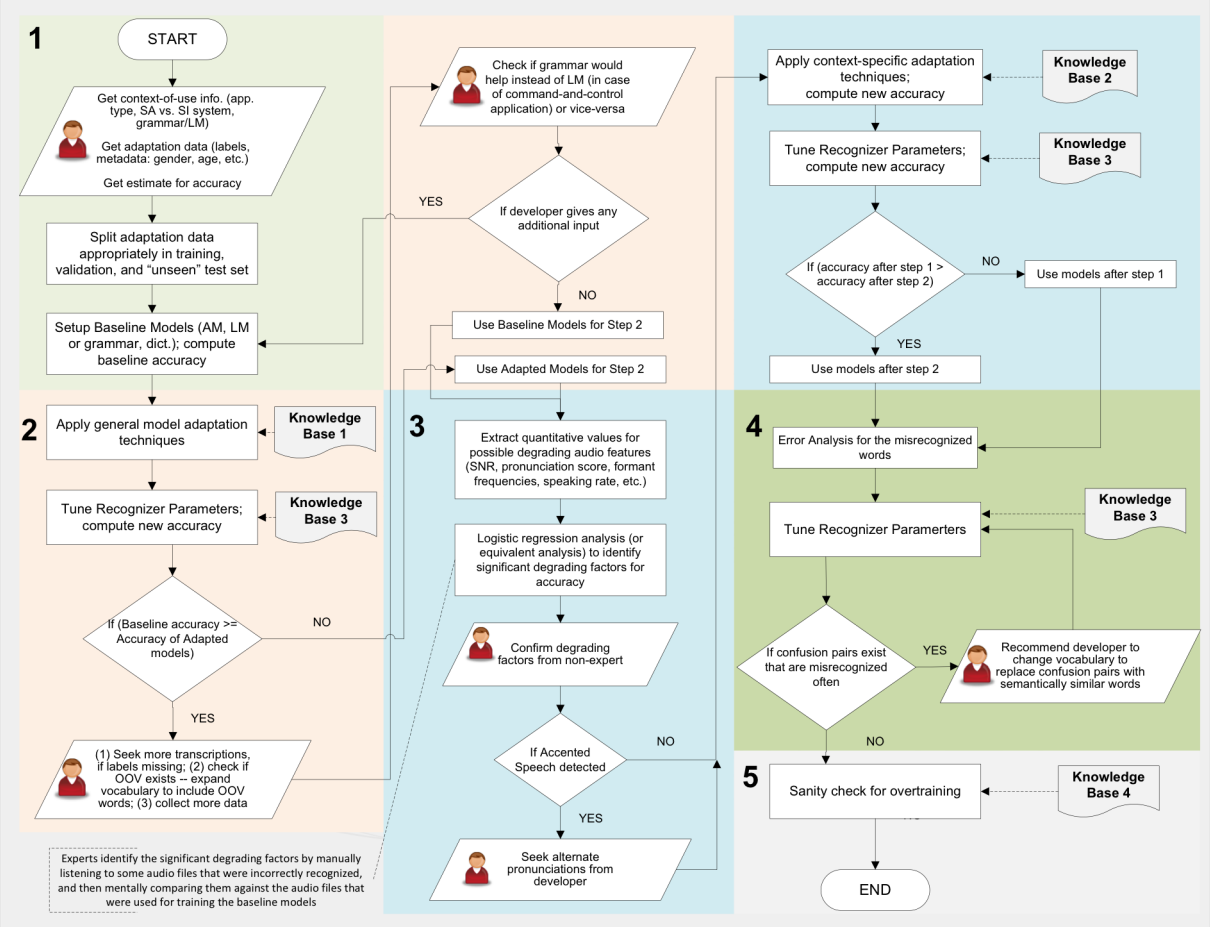


Figure 5.1: Flowchart to depict a typical adaptation process for a speech recognizer, as amalgamated from the interviews of speech experts. Possible interactions of the expert developer (or ultimately, the toolkit) with the non-expert SUI interface designer are denoted with a user icon.

Finally, a further advantage of splitting the above process into multiple components (check, adapt, optimize, tune, integrity check) is that most experts have specialized domains of expertise, even within speech recognition. These domains align well with the above

components, e.g. a data analyzer or a staff member usually focuses on data cleansing activities, an adaptation expert focused on the acoustic model transformations discussed above, an optimization expert or an speech recognition analyst focuses on identifying the degrading issues and performing the corresponding normalizations or optimizations, etc. Therefore, when it comes to verification and expansion of the current knowledge base, each expert can focus on their domain of expertise and ensure that the knowledge base corresponding to independent components above are updated and consistent with the state-of-the-art methods.

5.4.2. Characteristics of Knowledge Base

In addition to the interviews, we also reviewed over 15 publications on speech adaptations and error analysis, e.g. see [Bali *et al.*, 2013; Goronzy *et al.*, 1999; Lee *et al.*, 1992; Luce *et al.*, 1998; Shinoda, 2011; Woodland, 2001; Yee *et al.*, 2012] to further understand techniques that experts use. Based on these, our current knowledge base covers the following variations spanning over 80 rules:

1. Application: isolated words, dialogue system, or dictation.
2. Recording Device: external headset, telephony, etc.
3. Speakers: single, multiple, speaker-independent or not.
4. Vocabulary Size: tiny (<100 words), small (>100 words), medium (>1000 words), large (>10000 words).
5. Vocabulary Type: expressions, technical, conversation, etc.
6. Adaptation Data Size: small (1hr), medium (5hrs), large (10hrs), very large (>50hrs).
7. Availability of Transcripts: exists for the entire set, partial set, or none.
8. Noise Conditions: quiet room, street, competing talkers, etc.
9. Speaker Factors: age, gender, emotion, dialect, etc.

Note the definitions of tiny, small, medium, large, and very large for some of the above characteristics have been noted after consulting the experts we interviewed. It is possible that experts in other groups or professions might have a different definition, however, the rules instituted in the current knowledge base follow the terminology listed above.

5.4.3. Expert System and Forward Chaining Inference Engine

An expert system typically consists of three components as follows:

1. Knowledge Base: contains specific facts about the expert area and rules that the expert system will use to make decisions based on those facts.
2. Inference Engine: an engine that process existing facts using the rules instituted in the knowledge base.
3. Observation Set: a set of current facts or conditions.

For a rule-based knowledge base, the inference engine can either be implemented with forward chaining or backward chaining. In forward chaining, the processing starts from given observations, and rules are fired based on whatever observation are true. Firing these rules may result in finding more observations (which are then added to the observation set), or finding directions for the system to execute. This process continues until all rules have been processed, i.e. have either fired or rejected. As an example, consider the following situation:

Observation set = {O1, O2, O3, O5}
Knowledge Base = {rule #1: If O1 and O2, then O4;
rule #2: If O4 and O5, then D1;
rule #3: If O5 and O6, then D2}

In the first round, rule #1 is fired as O1 and O2 exist in the observation set, and O4 is added to the observation set. In the second round, rule 2 is fired, because the revised observation set now contains both O4 and O5, and direction 1 is executed.

In backward chaining, the processing starts from selecting a rule and regarding it as a problem to be solved. Such procedures are often called goal driven inferential processes. For example, the inference engine might be given the goal to "find the all observations that need a specific technique to be used" and would work back from there, asking questions as necessary to confirm. For instance, in the above example, a question could be asked, “do we have all the necessary conditions to perform direction D2”. The inference engine would select rule #3, and check if D5 and D6 exist in the observation set, or if any other rules could be selected that have D5 and D6 as its result. In our case, the inference engine is as such forward reasoning, as the requirement is to identify appropriate techniques (directions) given an observation set.

5.4.4. Conflict Resolution

On various occasions, we received conflicting recommendations from the experts. For instance, such conflicts arose for the initial setup condition, e.g. when to start with an existing baseline or train a new acoustic model. In the knowledge engineering literature [Negnevitsky, 2005], three approaches have been suggested to deal with conflicts while formalizing expert knowledge:

- Fire the rule with the highest priority. The priority can either be assigned as a number or a just the order of rules in the knowledge base can denote priority.
- Fire the rule that’s most specific. For instance, if three rules exist in the knowledge base, “*IF A and B, then X*”, “*IF A, then Y*”, and “IF B, then Z”, and observations A and B both exist, then fire the first rule as it is more specific. This idea is based on the assumption that a specific rule possesses more information than a general rule. It also takes into

account that somehow the recommendations will change when the combination of observations are present together, instead of independently.

- Fire the rule that uses the most recent observation. This method assumes that the observations are time dependent and that they have time stamps.

In the context of the knowledge base for speech recognition, we instituted conflicting options as alternates, and ranked them based on the number of experts that recommended the option. In other words, we assigned priority rankings based on the "the number of experts that recommended each option", and fired the rule that had the highest priority. As such, the other two approaches of firing the rule that's more specific, and firing the rule with most recent observation did not apply for our knowledge base as the observations were neither dependent on each other, nor had time dependence. Furthermore, to be able to learn from its mistakes, we allowed the inference engine to update the priority rankings in the knowledge base of competing options based on results of actual test runs. Here's how the system would work: in the event a rule with multiple options gets triggered, the knowledge base would first recommend the top most option; and if the non-expert developers were unsatisfied with accuracy, they could query the knowledge base again for an alternative, if any existed. Based on the results of what actually worked, the developer could provide feedback to the knowledge base that would update the ranking of the alternatives.

5.5. Datasets

We used the following two datasets for our evaluation:

5.5.1. "Seen" Dataset A: Indian Children's Speech

Dataset A – also used during the knowledge elicitation process – comes from case study #1 [Kumar et al., 2012]. In total, it comprises of 6250 single-word English utterances from 50 rural Indian children (~2 hours of speech). The speakers are equally split across gender and grades 4-5. Each child spoke 125 words selected from a set of 325 words, ensuring equal coverage for all words. The audio recordings were made on Nokia N810 using a close-talk microphone in out-of-classroom, noisy backgrounds. In our experiments, we set aside data from 10 speakers (5 males) as test set, and the remaining available as tune set (corresponding to the data that experts will receive from the field researchers).

5.5.2. "Unseen" Dataset B: CMU Kids Corpus

Dataset B, popularly known as "CMU Kids Corpus" [Eskenazi *et al.*, 1997], comprises of English sentences read aloud by 76 children (24 males). In total, the dataset has 5180 recordings where each recording represents one sentence. The recordings were made in a classroom during school hours, and there are some files with background noise such as students fiddling with the equipment. There are two types of speakers: "good" readers (44) and "poor" readers (32). For

the latter, the speech has some typical reading errors that children make when not confident about a text. In our experiments, we set aside data from 16 "poor" speakers (8 males) as test set, and remaining available as tune set (corresponding to the data that experts will receive from the field researchers).

5.6. Data Annotation

Labeling each audio file on its acoustic and behavioral properties is critical to understand the reasons of poor (or conversely, good) speech recognition performance. We annotated our data on the following features that correlate with popular reasons of recognizer failure.

1. **Pronunciation score** measures the quality of user's pronunciation on a scale of 1-10, in comparison to a native English pronunciation. To calculate the score, we followed the pronunciation scoring strategy as in [Ronanki *et al.*, 2012]. In this strategy, we first force-align the audio file with its transcript to obtain a phone-segmented file. Next, we calculate several aggregated statistics of each phone from these files and compare them to exemplar statistics of native English pronunciations from TIMIT data [Garofolo *et al.*, 1993] to get a pronunciation score.
2. **Signal-to-noise ratio (SNR)** measures the strength of the signal with respect to the background noise. In other words, it measures the quality of the audio file, independent of the word uttered. This account for variability in the microphone's position while recording, or background noises. We employed the "direct space search" algorithm implement by the widely recognized NIST Speech Quality Assurance Package [nist-spqa].
3. **Fundamental frequency (F0)** of voice is the lowest frequency at which human vocal cord vibrates. **Formant frequencies (F1-F3)** are its spectral peaks. These were calculated using Praat [praat], and measured in Hz.
4. **Speaking Rate (SPR)** is the speed at which the user talks, typically measured in phonemes or syllables per second. It is calculated as the average time for the user to say one phoneme

$$SPR = \frac{number\_of\_phonemes}{start\_speech\_signal\ (ms)\ - end\_speech\_signal\ (ms)}$$

5. **Intensity** is the sound power per-unit area and is a measure of how loud the sound is. It was also calculated using Praat [praat].

5.7. Evaluation Experiments

After annotating our datasets, the first question we sought to answer was whether the Knowledge Base (KB) correctly modeled experts' knowledge and whether we could quantitatively identify the degrading factors that the experts recognized through manual inspection (in step 3 of the ChAOTIC process) for the same dataset as used during contextual interviews, i.e. Dataset A. Figure 5.2 summarizes the sub-steps involved in identifying the degrading factors in the speech recognition results.

Figure 5.2: A visualized representation of the sub-steps used in identifying the significant degrading factors in speech recognition results, as in Step 3 of the ChAOTIC process framework.

5.7.1. Experiment 1: Dataset A – Old Experts vs. Knowledge Base

To do so, we compared the recommendations made by the KB with those made by the "old" five experts (referred later as E1 through E5), and how those translated into accuracy gains. In other words, we wanted to assess its internal validity. The objective was to build a speaker-independent isolated word recognizer for 325 words in dataset A.

	Univariate		Multivariate	
	β	***p*-value**	**β**	***p*-value**
Pron. Scr.	0.07	0.05*	0.19	0.03*
SNR	0.24	0.01*	0.02	0.74
SPR	0.05	0.04*	-0.44	0.01*
F0	0.62	0.01*	0.19	0.14
F1	0.34	0.09	0.00	0.98
F2	-0.05	0.50	0.01	0.85
F3	-0.05	0.53	0.02	0.79
Intensity	0.70	0.06	0.69	0.02*

Table 5.1: Results from linear regression. β indicates the coefficient of regression and p-value indicates its statistical significance. (*) means p<0.05.

Table 5.1 highlights the results from the univariate and multivariate linear regression of the several degrading factors that we investigated. Non-native English accent (pronunciation score) and speaking rate turned out to be significantly correlated with accuracy for Dataset A in both univariate and multivariate regression, implying that they significantly impacted accuracy and their impact did not suffer from interaction effects with other factors respectively. At the same time, E3 and E4 recognized accent as an issue as well from their manual inspection, and recommended "Dictionary Adaptation" i.e. adding pronunciation variants to the dictionary as a potential solution (as in Table 5.2). E1 also acknowledged accent as an issue, but mentioned that adding pronunciation variants may not result in much accuracy gains, as general adaptation techniques like MLLR and MAP (in step 2 of the ChAOTIC process) had already been done. Instead, he was working on a new accent modeling technique, but since it was not published yet,

we did not include it in our analysis. None of the experts identified speaking rate (SPR) as an issue. None-the-less, we considered both accent and SPR for our next step.

Table 5.2 summarizes the recommendations from the experts and the KB, and the corresponding accuracy gains (or conversely, reductions in error rates). Overall, the KB outperformed all experts on test set for Dataset A. One point of difference came from starting with an existing acoustic model (AM) such as WSJ and then adapting it for the tune set using general adaptation rules such as MLLR and MAP (E1-3), or training one from scratch (E4-5 & KB) using the tune set. This decision was part of step 1 in the ChAOTIC process. E1-3 had reasoned that the amount of tune data available for training a new AM is very small, and so starting with an existing AM is best; whereas E4-5 had reasoned that the characteristics of the data were different from any available AM, and given that it's a small vocabulary recognition task, it would be best to train a new AM. While designing the KB, we had instituted both the options as alternatives for small vocabulary, low-resource situations. Given that 3 experts had recommended to adapt an existing model, the KB first recommended that option, but once the low accuracy was noticed, it switched to the next best alternate i.e. train a new acoustic model. In addition, after "seeing" the results, it updated the ranking of the two alternatives.

Optimization Techniques	**Experts**					**KB**
	E1	**E2**	**E3**	**E4**	**E5**	
1 - Baseline: existing AM	94.7	94.7	94.7			94.7
1 - Baseline: train AM				25.4	25.4	25.4
2 - MLLR	77.3	77.3	77.3			
2 - MAP	78.3	78.3				
3 - VTLN		76.9	76.9		24.9	
3 - Dictionary			71.7	22.2		22.2
3 - Frame-rate						20.8
Final	**78.3**	**76.9**	**71.7**	**22.2**	**24.9**	**20.8**

Table 5.2: Word error rates (in %) for test set in Dataset A. The numbers in the first column are the step nos. from ChAOTIC process framework.

5.7.2. Experiment 2: Dataset B – New Experts vs. Knowledge Base

Our second evaluation focused on assessing the generalizability of the KB for an unseen dataset B, and comparing its recommendations against new experts (E6 & E7). In other words, we wanted to assess its external validity. We asked E6 & E7 to develop a speaker-independent speech recognizer for "poor" speakers, as in Dataset B. Table 5.3 summarizes their recommendations and the error rates comparison with those from KB. The KB correctly identified data cleaning issues such as half words (E6-7) and spelling errors (E7), and

recommended that we train our own acoustic model (E7). E6, however, felt that since we had access to data from "good" readers, we should first train a "high-quality" model on good speakers, and then use it to find the alignments on the data from poor readers. This technique, known as model bootstrapping, is useful in low-resource situations, especially when you have access to a high-quality recognizer. Since we did not have a high-quality recognizer when the KB was developed, this technique was not covered by it. None-the-less, it is simple to add because we have a clear definition of its use, i.e. "presence of high-quality recognizer", and "low-resource situation". For the next optimizations, the regression correctly pointed SNR (E6) and SPR (E7) as the degrading factors.

Techniques	**E6**	**E7**	**KB**
1 - Add half words to dictionary	✔	✔	✔
1 - Correct spelling errors		✔	✔
1 - Baseline: train AM		56.1	56.1
1 - Baseline: train AM + model bootstrapping	42.2		
3 - CVN (SNR)	40.9		54.3
3 - Frame-rate (SPR)		45.2	52.2
Final	**40.9**	**54.8**	**52.2**

Figure 5.3: Word error rates (in %) for test set in Dataset B. The numbers in the first column are the step nos. from ChAOTIC process framework

5.8. Discussion and Next Steps

From the above results, we can make the following observations:

- With respect to step 1 and 2 of the ChAOTIC process, for both datasets, the Knowledge Base, in most cases, recommended the better of the techniques recommended by all the experts. When there was a conflict between experts during knowledge elicitation, the knowledge base picked the technique with the highest ranking, i.e. the technique that was most recommended during knowledge elicitation. However, in case, the chosen technique had poor results, the developer-users could query for the next best alternative, and also provide feedback if the alternative worked better in the context of their dataset. Unfortunately, there's no exact definition of "poor," as the threshold of acceptable baseline recognition is dependent on the type of task (dictation vs. conversation vs. isolated words), size of the vocabulary, and many other factors. However, the feedback provided by the developer-user helps the knowledge base adjust its ranking of different alternatives and over-time provide higher quality recommendations first.
- With respect to step 3 of the ChAOTIC process, for dataset A, the Knowledge Base identified additional degrading factors that the experts missed out (i.e. SPR), in addition to those that the experts identified from their manual inspection (i.e. accent). Corresponding

adaptations for both the degrading factors led to improvements in the accuracy indicating that the knowledge base can not only mimic expert's intuition, but also, in some cases, surpass it by a more systematic error analysis. For dataset B, E6 and E7 identified two different degrading factors. The KB, through its regression analysis, identified both of them, which both led to accuracy gains. In future, a more comprehensive evaluation with other datasets would of course be needed to establish these claims further.

- For dataset B, E6 recommended a technique that was not initially covered by the knowledge base, but was significantly better than the alternatives that the KB had. With researchers pushing the boundaries of speech recognition, such situations are likely to arise in the future as well as new techniques become available, or old techniques become completely obsolete. Fortunately, by its rule-based design, the knowledge base can be easily updated to include newer techniques by "expert researchers," who would be maintainers of this knowledge base. This aspect of updating rule-based knowledge base has also been demonstrated in the knowledge engineering literature [Richards, 2009].

In this chapter, our aim was to understand the tacit process that experts use to optimize a speech recognizer, and formalize it in a rule-based knowledge base. Through interviews we experts, we formalize the tacit process and presented its key six steps (also abbreviated as ChAOTIC). We also presented the design and characteristics of a rule-based knowledge base that models experts' understanding of speech recognition issues. Further, we assessed if with the help of this knowledge base (KB), we can automatically recommend appropriate optimization steps. Our results show that the KB can successfully predict optimization techniques for previously seen acoustic situations during the interviews or the papers reviewed, both in seen and unseen datasets. Moreover, by the nature of its rule design, it can be expanded to incorporate additional insights on specific acoustic and language contexts by other experts [Richards, 2009].

In the next chapter, we present the design and development of SToNE: Speech Toolkit for Non-Speech Experts, which integrates this knowledge base in a infrastructure over the web and in a Virtual Machine, so that the KB can improve from the experiences of test runs of many potential users. We also discuss the design and development of several error analysis tools that form a key part of SToNE. In Chapter 7, we evaluate SToNE with semi-experts and non-experts on the task of building a children's LVCSR.

6. STONE: SPEECH TOOLKIT FOR NON-EXPERTS

STONE: **S**peech **To**olkit for **N**on-**E**xperts is a toolkit that enables non-speech experts to easily develop accurate speech recognizers for any user or acoustic situation, with minimal requirements for expertise in speech recognition. This chapter relates to Research Goal 2, which concerns the design and development of STONE. In particular, the sub-goals in RG2 are:

- **Sub-goal 1:** Enable a fine-grained understanding for non-expert developers of *why* a speech recognizer is not working,
- **Sub-goal 2:** Provide guidance to the non-experts on *what* they can do to make it work. This includes both optimizations to the underlying recognizer and the user interaction style. In some cases, it will also automatically perform the recommended optimizations.
- **Sub-goal 3:** Enable performance visualization of competing setups to understand the tradeoff's involved in each setup.

These sub-goals and their actual realizations in the toolkit are further discussed in Section 6.2.

6.1. Related Work: Toolkits and Error Analysis

6.1.1. Toolkits in Machine Learning

Machine Learning deals with a general class of problems such as speech recognition or image classification. Given the success of machine learning (ML), in general, for effectively solving hard problems, researchers have built numerous toolkits to support the development of ML systems. It is therefore important to understand and relate to this prior work.

We classify the current approaches into three broad categories: *general-purpose algorithms* toolkits, *domain-specific* toolkits, and *interactive visualizations* toolkits. Algorithms toolkits are general-purpose tools that provide APIs for using popular machine learning algorithms. These APIs can be used for any domain and do not enforce any domain-specific knowledge. On the hand, domain-specific toolkits focus on a single application domain, e.g. computer vision or image recognition, and encode knowledge about that domain to help build systems specifically it. Interactive visualizations toolkits focus on analysis and refinement by enabling exploration of

data or results to improve a system further. Most of the interactive visualizations toolkits are general-purpose tools that do not encode any domain-specific methods of analysis. We discuss each of these below in more detail below.

6.1.1.1. General-Purpose Algorithms Toolkits

Implementing a bug-free ML algorithm from scratch is hard, and this implementation usually forms an important barrier to using machine learning. Many APIs provide easy to use interface for ML algorithms [Demšar *et al.*, 2004; Witten *et al.*, 2005], and a number of toolkits have been developed that provide a library of such APIs, e.g. Weka [Hall *et al.*, 2009], or scikit [scikit]. At the same time, algorithms are only one of the many different factors that determine the behavior of a ML system. Effectively training a ML model also requires effectively working with data. Often the data is error-prone, inconsistent, skewed, or even redundant. Identifying and removing these irregularities is also an integral part of the development process. Moreover, developers need to understand the relationship between data and the trained ML model, for example, in a speech recognition system, if the training data is skewed and has most audio files for a specific type of noisy environment, then the trained model is bound to fare poorly in a clean environment or even other types of noises. Gestalt [Patel *et al.*, 2010] tries to address this issue by providing support for both APIs and support to collect and analyze data.

Other disadvantages of general-purpose algorithms toolkits is that although they reduce the threshold of usage of ML algorithms from an "ML expert" to a "developer," it still requires a working understanding of the algorithm to be able to properly fine-tune and optimize the algorithm for any given dataset. There is no guidance to simplify this step. Additionally, these tools often require programmers to convert their data into a common format. Programmers have to write the data-processing code somewhere else, but it is hard to connect the raw data back to the model after converting it into a common format. General-purpose tools need to support the inspection of raw data in order to help programmers detect errors in their data and come up with new features.

6.1.1.2. Domain-Specific Toolkits

Domain-specific tools provide greater functionality for a specific domain, but do not generalize well to other domains. In other words, they trade flexibility for many algorithms in favor of enhanced functionality for a specific domain. Because they focus on a specific domain, they are also able to limit the types of input that the toolkit might expect, and hence may remove problem of inconsistencies in input data format across domains. This becomes better suited for "non-experts" or developers with little knowledge of the data normalization or conversion techniques.

Crayons is one such domain-specific tool that supports image segmentation [Fails *et al.*, 2003]. A developer captures an image and colors the regions that correspond to different segments. The system learns a model from these labeled pixels, and the developer analyzes the model's performance by applying it to new images and overlaying the results on those images. The designer iterates by correcting model mistakes, thus providing new data for the classification pipeline. Crayons achieves this ease of use by limiting flexibility. Input is limited to providing more training examples, and analysis is limited to looking at classification results overlaid on images. Developers cannot access other information that might help them iterate (e.g., attribute values). Domain-specific tools have been created for a variety of problems, including computer vision systems [Maynes-Aminzade *et al.*, 2007], simple sensor-based recognizers [Hartmaan *et al.*, 2007], and interactive concept learning in image search [Fogarty *et al.*, 2008].

Both general-purpose tools and domain-specific tools are complimentary. It is almost certain that a highly specialized tool will be more effective for its particular problem. However, general tools can support problems for which domain-specific tools have not yet been developed. Further, distilling general mechanisms can inform the design of domain tools by allowing a focus on domain-specific extensions instead of reinventing general mechanisms.

A few toolkits are somewhere in between a domain-specific and a general-purpose toolkit. For instance, Hidden Markov Model Toolkit (HTK) is a domain-specific toolkit for speech recognition, but is used for a larger class of pattern recognition problems, such as character recognition, DNA sequencing, or speech synthesis.

6.1.1.3. Interactive Visualization Toolkits

There is another class of toolkits that shifts focus from implementation of algorithms or development of domain-specific ML models to further refinement. For instance, there are toolkits that enable indirection between results and actions, i.e. the ability to directly interact with the output of the model in order to specify preferences for the type of results desired. As an example, a programmer training a digit recognition system may observe that the system is unable to distinguish between fours and nines. They may prefer to make errors on other digits higher in order to improve the performance of fours versus nines. Traditionally, there hasn't been a way for programmers to directly specify these preferences.

There is recent work that enables the developers to directly interact with the output of the model in order to improve performance. For instance, EnsembleMatrix [Talbot *et al.*, 2009] is one such tool. An ensemble is a meta-classifier that combines results from many different models. EnsembleMatrix trains an ensemble for large multi-class problems. It then provides an interactive confusion matrix view that can be used to steer the accuracy of the classifier. The

programmer steers the classifier by grouping regions of the confusion matrix. ManiMatrix [Kapoor *et al.*, 2010] builds on this idea by having programmers interact directly with confusion matrix visualization to help guide the model. Developers can indicate preferences by clicking on cells in the matrix. These preferences are translated into a cost matrix. The underlying machine learning algorithm then optimizes this cost matrix in order to automatically guide the development of the model.

6.1.2. Error Analysis in Speech Recognition

In speech recognition research there are typically two approaches for identifying error sources [Van Ess-Dykema *et al.*, 1998]. At one end, we have the "automatic" approach that is typically adopted by the engineering community. In this approach, the focus is on a single metric, optimally on a single number such as word accuracy or perplexity, which can be calculated automatically. Usually, this number is calculated for the entire test dataset, and the focus is on optimizing the single metric for whole dataset. On the other end of the spectrum, linguistic experts adopt the "manual" approach. Here, the focus is on comparing decoded results of the recognizer with human transcripts to determine specific properties or patterns of the errors produced. This approach is tedious as experts comb through every audio file, one by one.

Neither approach is ideal. The engineering experts' approach is hard for generating detailed insights into why the recognizer is failing as they focus is on single metric for the entire dataset.

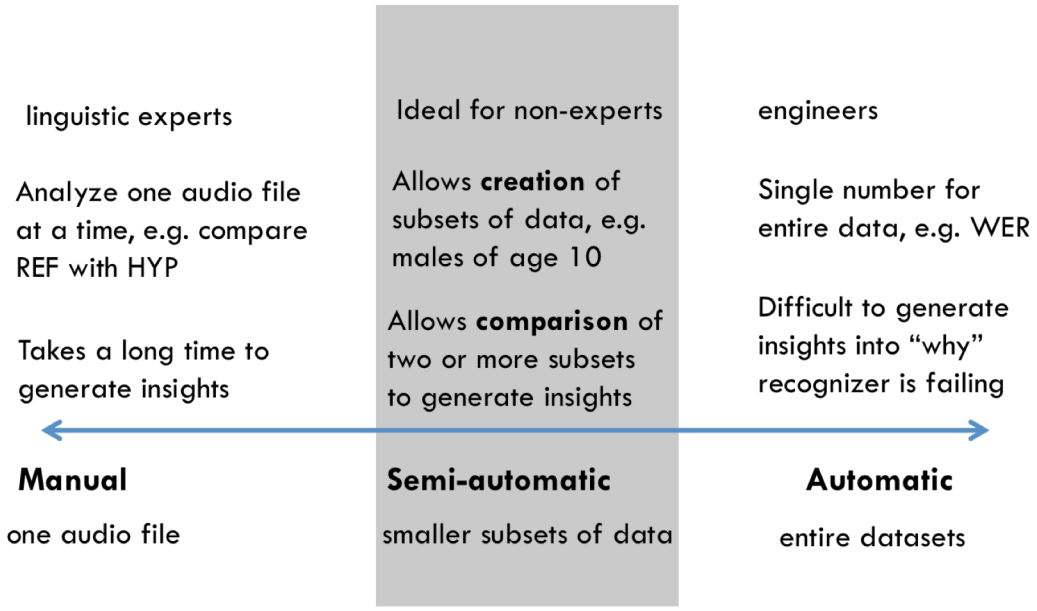


Figure 6.1: Error analysis in speech recognition as a continuum. Both automatic and manual analysis are suited for experts, albeit different types of experts. Semi-automatic analysis lends itself better for non-experts because it allows for creation and comparison of smaller, meaningful subsets of larger datasets, thereby assisting in discovering what parts of systems are underperforming, and why they are not performing as well as other subsets of data.

The linguistic experts' approach can take a significantly long time and reading the transcripts and assigning error sources can be tedious and confusing. Additionally, the evidence generated is often small and may be irrelevant.

In thinking about the design for SToNE, we argue that there's a sweet spot that exists between these two approaches (as shown in Figure 6.1), which enables semi-automatic error analysis. In this approach, developers are able to easily **create** meaningful subsets of the dataset, **visualize distributions** of specific parameters for those subsets to uncover regions of error, and **compare** two or more subsets to generate insights, e.g. by comparing a subset of files with accuracy greater than 70%, and other less than 30%, the developers can understand what's different between the two sets and discover reasons (SNR, SPR, accent, gender etc.) of why certain files are not well recognized. In the literature, some work has already been done on identifying typical types of errors in speech recognition system, and potential reasons. However, independent analysis must be done for each speech system that's built, as reasons of error may vary tremendously from one system to another. In our work, we aim to leverage previous findings in speech recognition error analyses by enabling developers to perform semi-automatic analysis on various types of errors and their reasons.

6.1.2.1. Types of Errors

State-of-the-art speech recognition still produces a substantial number of errors. Knowledge of the types of errors can help direct efforts towards removing the sources of errors. However, uncovering such errors, even despite the availability of the recognition results is a challenge in and of itself [Goldwater *et al.*, 2010]. At a coarse level, error rate in speech recognition is often divided into three subparts: insertions (I), deletions (D), and substitutions (S). These are obtained by aligning the transcript of the audio file with that of the hypothesis from the speech decoder. For instance, consider the phrase "It's very hard to recognize speech," and its popular output, "It's hard to wreck a nice beach."

String alignment algorithms, e.g. Levenshtein's distance [Levenshtein, 1966] aligns the strings as:

Transcript:	It's	fun	to	recognize		speech
Recognizer output:	It's	fun	to	wreck	a nice	beach
Type of error:				S	I I	S

As the reader would notice, insertion errors are those when an additional word is added to the recognizer output (*a* and *nice* in the above example). Substitutions errors are those when a word in the transcript is replaced by another word in the output (*wreck* and *beach* in the above

example), and deletion errors are those when a word is omitted from the output (none in the above example). The overall word error rate is a sum of all three types of errors.

Name of Error	Example	Explanation
Pronunciation	**Four egos** (**before he goes**)	Caused by mismatch between the user's accent and the accent that the recognizer is trained on.
Homonyms/ Phonetic confusion	She has a bone spur at the **see three** (**C3**) vertebral level	Caused by similar sounding words getting higher precedence in the decoding process, i.e. recognition is worse for words that are phonetically similar to many other words than for highly distinctive words (Luce et al., 1998)
Suffix	He has eat (eat**en**) his meal	These errors usually occur when part of a word is silent in the audio file, and the language model does not prioritize the correct form of the word over the acoustic model's output.
Dictionary	He wanted to visit **be as her** (**Bilaspur**)	Some words, e.g. names are absent from the vocabulary. These are referred to as "out-of-vocabulary", and are a common source of substitution errors, especially in free-form conversation speech.
Spelling	He did not want to take a **favour** (**favor**)	American English vs. British English. Sometimes the vocabulary list and the dictionary itself would have spelling errors.
Word boundary	It's hard to **wreck a nice beach** (**recognize speech**).	Incorrect word boundaries can often lead to incorrect recognition, and error propagation. For instance, here, /s/ from "speech" got attached "nice" instead of "speech".

Table 6.1: Types of substitution errors along with their examples and explanations. In the examples, the words in the brackets indicate correct transcripts.

A simple understanding of the proportion of these errors often leads to tuning recognizer parameters. For instance, if there are too many insertion errors, the speech recognizer developer can increase the insertion penalty, and reduce the number of words being outputted in the decoded output. This, however, comes at a cost of increasing deletion errors. Therefore, the developer must find the right balance between number of insertions and deletions for their recognizer, which usually corresponds to about the same number of insertions and deletions. However, there's no predetermined rule for arriving at an optimal insertion and deletion penalty.

In terms of substitutions, there are many types of confusions that can be generated. Some examples are presented in the table below

6.1.2.2. Reasons of Error

Previous research [Goldwater *et al.*, 2010] has shown that prosodic and lexical variations across speakers or their utterances, or other external factors such as noise are the dominant reasons of speech recognition error. Yet expert researchers find it extremely difficult to incorporate this deeper analysis in their everyday analysis techniques because of challenges involved in extracting feature values for each of the prosodic and lexical variations. Even when they do, their analysis is either limited to a subset of the parameters (which they guess to be most relevant), or is done for the entire dataset as a whole. In the case of the latter, it might be difficult to identify the exact reason of error, as the impact of a specific variable might be withered out when averaged across all test speakers, especially in cases when it affects only a subset of the data. We argue that in the proposed semi-automatic approach, both experts and non-experts will be able to identify smaller subsets that correspond to one of the following reasons of error.

Speaking Rate

We speak in different modes of speed, at different times. If we are stressed, we tend to speak faster, and if we are tired, the speed tends to decrease. We also speak in different speeds if we talk about something known or something unknown, and hyper-articulate when someone does not recognize us in the first attempt, also known as the Lombard effect [Junqua, 1993]. In general, the speaking rate tends to have an impact on the accuracy, especially in free-form dictation or conversational recognizers. Fast speech is associated with higher error rates [Siegler *et al.*, 1995; Fosler-Lussier *et al.*, 1999; Shinozaki *et al.*, 2001]. At the same time, very slow speech has also been found to correlate with higher error rates [Siegler *et al.*, 1995; Shinozaki *et al.*, 2001]. However, in some other studies, results for speech rate were ambiguous: faster utterances had higher error rates in one corpus, but lower error rates in the other [Hirschberg *et al.*, 2004]. The exact correlation needs to be examined for each dataset individually, and not just that, it needs to be examined for subsets of the data individually, as different portions of the test data can have different speaking rates.

External Noise

When speech recognition systems are used in noisy environments, there performance tends to drop drastically in comparison to quiet, lab environment. For example, an isolated word recognizer trained on real speech gives 100% accuracy with clean speech, and this dropped to 30% when used in a car travelling at 90kmh [Lockwood *et al.*, 1992]. Other types of noise, e.g. white noise, cross talk, etc. also have severe degrading effect on the recognition accuracy.

Speaker Gender & Age

Adda-Decker *et al.* (2005) demonstrated that both French and English ASR systems had more trouble with male speakers than female speakers, and suggested several possible explanations, including higher rates of disfluencies. Somewhat contradicting, females had a higher error rate than males in a study conducted by Vergin *et al.* (1996) who concluded that higher fundamental frequency of females voice leads to more errors. Moreover, error rates are even higher for children (in comparison to adults) due to variabilities in vocal tract length, formant frequency, pronunciation, and variable speech [Hagen *et al.*, 2007; Eskenazi, 2009].

Regional and Social Dialects

Dialects are group related variations within a language, such as specific pronunciation styles, or specific vocabulary and grammar based on the geographical area the speaker comes from [Holmes, 1992]. In many cases, these dialects are considered as "another language" in ASR, due to the large differences between two dialects, and completely new pronunciation dictionaries and acoustic models are trained for speakers from different dialects.

Word Length (in phones)

In Shinozaki *et al.*'s (2001) analysis of a Japanese ASR system, word length (in phones) was found to be a useful predictor of error rates, with more errors on shorter words.

Misrecognized Turns

In Hirschberg *et al.*'s (2004) analysis of human–computer dialogue systems, misrecognized turns between speakers were found to have (on average) higher maximum pitch and energy than correctly recognized turns, and misrecognizing the first word at the turn led to error propagation resulting in higher word error rate.

6.2. SToNE Design and Implementation

In thinking about the design of SToNE, our focus was on supporting an entire pipeline for developing a custom speech recognizer, i.e. setting up a baseline, implementing relevant optimization techniques, and analyzing errors along the way. While there are several parallels of building a speech recognizer to development of any other machine learning system, SToNE's design stands out from simple extensions of other ML toolkits in many ways:

1. First, we focus on providing *guidance*. While other machine learning toolkits simplify the machine learning pipeline by providing easy access to APIs and an environment to test various configurations, none provides guidance on *what* to do and *why*. Specifically: (i) we enable an understanding of ***why*** a recognizer is failing. This is done through error analysis and visualizations tools, which are not only specific to speech recognition, but also support

methods that experts' often use – either tacitly or explicitly – in understanding reasons of failure. (ii) We also provide knowledge of ***what*** to do for a specific problem. This is the work supported by the rule-based knowledge base developed in Chapter 5. In other words, we view SToNE as providing extensive *functionality* as in a domain-specific toolkit with the *flexibility* to pick and select appropriate algorithms pertaining to speech recognition – similar to as in an algorithms toolkit. It is also designed with the principles from an expert system to provide guidance to a "non-expert" to be able to pick the best optimization techniques or select optimum parameter values, something that neither of the ML toolkits have previously done.

2. Second, we do not leave *user interface design* entirely to the developer. Existing toolkits only focus on the development of the underlying recognizer, which is often the key, and leave the design of the user interface entirely to the human designer. SToNE encourages users to make appropriate changes to the design of the user interface by providing relevant guidance.

3. As a consequence of providing guidance and building a scalable knowledge base, SToNE employs a server-client architecture. The server contains modules that are applicable to all recognizers irrespective of what recognizer is developed, e.g. knowledge base or the feature extraction modules. The client – in this case a Virtual Machine – contains the actual installation of the speech recognizer (Kaldi), and SToNE's front-end interface to guide the developer in building a recognizer. Working in a virtual machine setup has several advantages, which we outline later in this section.

On the whole, SToNE's design is organized around three modules that are central to a speech recognizer's development: Baseline Developer, Error Analyzer, and Optimization Advisor. A typical use flow for SToNE is depicted in Figure 6.2. The three modules map to the six-step process described in Section 5.4.1. The baseline developer module contains rules from the "Data Setup" step, and enables development of a basic functional recognizer. The error analysis module contains methods and visualizations that experts use in understanding why a recognizer might be failing. The optimization advisor module guides the developer specific techniques that can help in improving the accuracy of the recognizer. The knowledge base module is a part of the optimization advisor module, and houses all the rules, except the ones concerning baseline recognizer development. Below we describe each of the sub-modules in more detail.

As mentioned previously, the implementation of SToNE follows a server client architecture. The server is responsible for hosting the knowledge base and feature extraction modules that are independent of the type of recognizer that needs to be developed. The client runs a virtual

machine, which is an independent environment running Ubuntu 12.04 LTS with access to 8 cores for performing speech recognizer's training and decoding. The advantages of running a Virtual Machine (VM) are several: (i) since many state-of-the-art speech recognizers (Kaldi, Sphinx, etc.) are currently actively developed for Linux based operating systems, Virtual Machines enable cross-platform operability. That is, a user with a Windows machine can also successfully run the latest version of the speech recognizer without partitioning the hard disk, as Virtual Machines are capable of virtually running any OS independent of the host OS. (ii) VMs enable easy sharing of progress and configuration setups to support easy debugging between experts and non-experts. Although, in this thesis, we don't specifically focus on evaluating the utility of VMs over other possible alternatives, we leverage this concept from other ongoing efforts, e.g. Speech Kitchen for distribution of speech recognition software [Metze *et al.*, 2013].

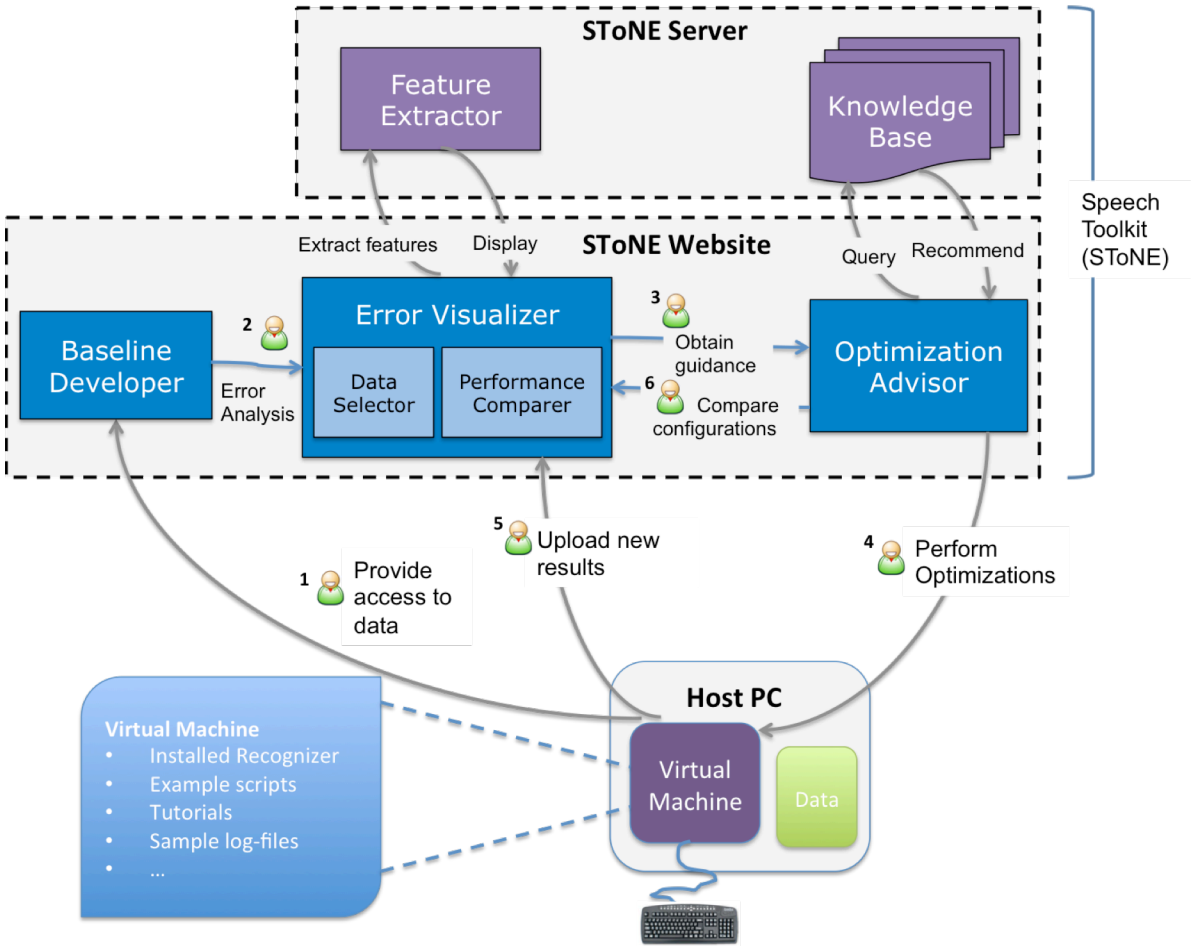


Figure 6.2: A process flow diagram of the developer's interaction with the modules of the toolkit. The numbers indicate the sequence of steps that the developer has to take, and the arrows indicate the direction of information flow. The developer uses a pre-configured virtual machine to conduct all experiments and interact with the website to avoid any hassles of recognizer installation, etc.

6.2.1. Step 1: Baseline Developer

The first step in building any speech recognition system is to develop a baseline recognition model. The baseline could either be an existing off-the-shelf recognizer that the developer might optimize for their use case, or start from scratch. In case of the latter, the developer should have access to a reasonable sized training dataset – for instance, for an LVCSR system, it should be about 75 hours of speech or more. The developer also needs to adapt an existing script from another example dataset to suite the training settings of their training dataset, e.g. if the audio files in their training dataset are long (>45 seconds), then they need to increase a parameter called "beam size" so that the recognizer can find proper time-alignments of the audio file with the corresponding transcripts. Experts usually use their intuition to set correct parameter values, and then iteratively modify those based on whether or not they were able to successfully develop a baseline. If, however, they are unsuccessful, they need to go through a large set of log files to understand why the baseline configuration failed, and then fix the appropriate parameters for the next try.

To solve this, SToNE provides a "baseline developer" module, which automatically adapts an existing script for the training dataset provided by the developer, and exposes all the necessary parameters in a graphical user interface to them. It also provides reasonable estimates of the initial parameters to the developer based on rules developed from expert interviews and knowledge base discussed in Chapter 5. In case baseline training is unsuccessful, it also parses the log files automatically, and guides the developer in changing the parameter values. In doing so, the level of guidance is restricted to suggestions of "appropriate ballparks" or the "direction of change", e.g. increase (or decrease) the current value, but supplying an exact value for the training is left to the developer. This is because finding an exact value that will work for sure is a research problem in and of itself, and experts have often relied on some intuitive guestimates, rather than a formal method of parameter estimation till date. We envision that providing guidance on what to change, which direction to change it, and a reasonable estimate are all valuable insights for a non-expert developer.

6.2.2. Step 2: Error Analyzer

The next step in the development process is to understand why the baseline recognizer may not be working well for the provided test set. Experts use their intuition and knowledge by looking at many sources of error to either *find a mismatch between training and test set* [2], e.g. presence of out-of-vocabulary words, accent, speaking rate, noise, etc., or by *looking at the output and making inferences from it*, e.g. high insertion rate, too many substitutions for a specific

[2] Since development of a speech recognizer is iterative in nature, it is possible that the developers overfit their training models to the type of test data they use. To avoid this problem, usually experts have a "development set", which they use as the test set during the development phase. A true test set is not seen until the end of the development cycle, and is used only to report the final performance of the speech recognizer.

phoneme, etc. SToNE partly automates this process, and partly provides analysis tools to help the developer in finding reasons of error. Below we detail each of them.

6.2.2.1. Automatic Regression Analysis

There can be many reasons why a recognizer fails, and several of them pertain to the acoustics of the testing set, e.g. pronunciation, speaking rate, noise, etc. To automatically identify the most significant distorting features, we run a linear regression analysis in the following way: first, we extract seven acoustic features for each audio file from the test set that are known to be major reasons of error in speech recognition literature [Goldwater *et al.*, 2010], namely pronunciation (p), speaking rate (r), sound quality (q), pitch (f0), and the first three formant frequencies (f1, f2, f3). Second, using accuracy for each audio file from the baseline recognizer as a dependent variable, we run a univariate and multivariate regression analyses to identify the most significant features impacting accuracy. These features are presented to the developer as system-generated reasons of error.

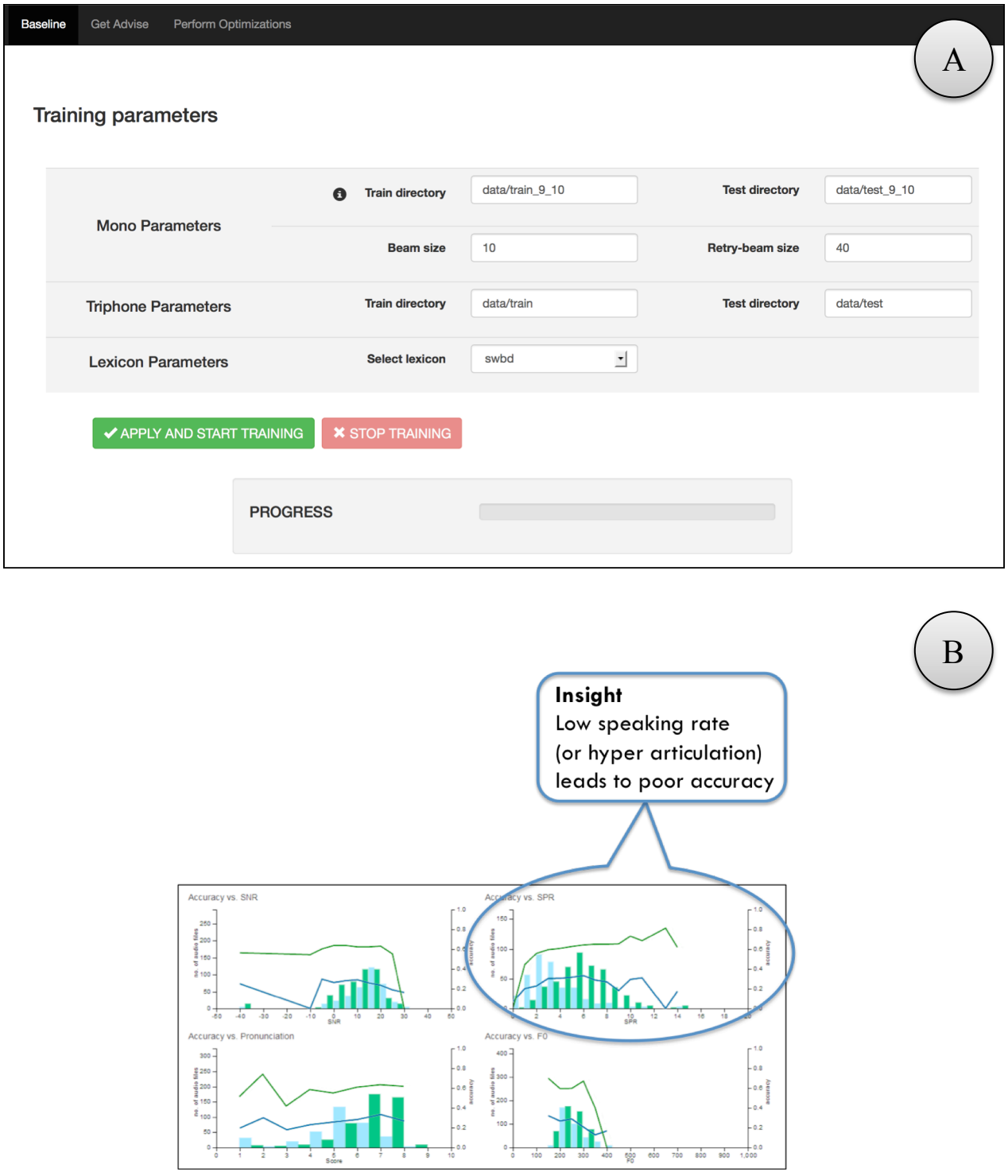


Figure 6.3: SToNE guides the developer in building a baseline and then in optimizing it. Development begins with (A) Baseline Developer module that helps the developer in properly setting up the data and build an initial recognizer. From here, the user could go to (B) Error Analyzer module, which helps in generating insights for reasons of recognition failure on a version of a recognizer, or directly to Figure 6.3(C) shown below to select optimizations techniques to problems that might be harming the recognition.

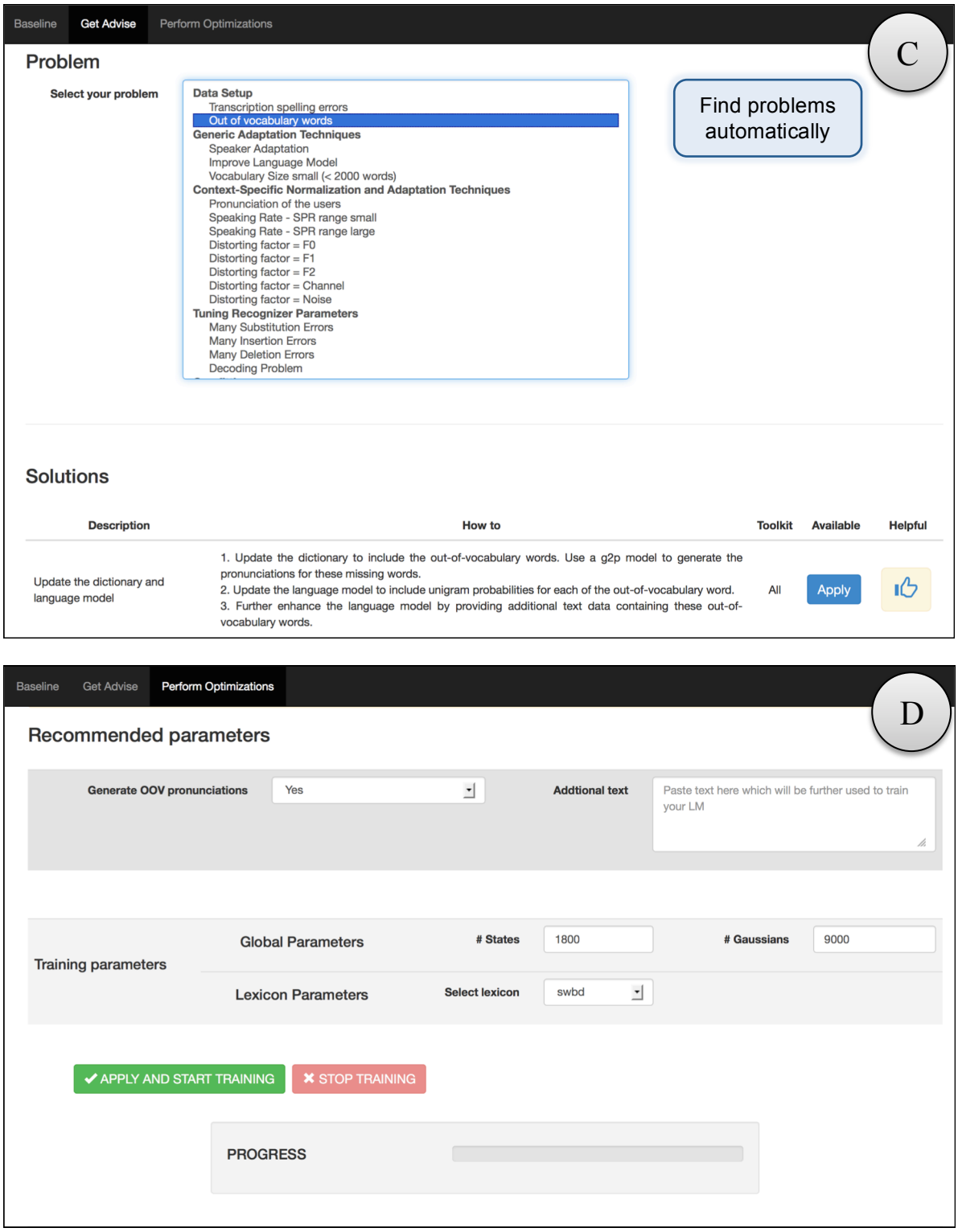


Figure 6.3: In (C) Optimization Advisor module, the developers can either manually locate problems and corresponding solutions, or let the module automatically identify problems and solutions that might be harming the recognition. Once a solution is appropriately selected, they can also get relevant implementation advice from (D) Perform optimizations module, which helps the developer in properly applying the selected solution, and retraining the recognition system.

6.2.2.2. Visualization and Comparison Tool

While the automatic analysis points to reasons of error based on regression analysis, there might be other issues that an "automatic" system might have missed. The visualizations tool helps the user analyze the test data on multiple factors to understand recognition error patterns, and generate insights into why the recognition might be failing. To have a finer understanding of why the recognition might be failing, it enables the user in doing four tasks:

(i) Understand the data distribution across the above variables – demographics (gender, speaker, age), prosodic (speaking rate, articulation, frequency, etc.), lexical (word length, etc.), environmental (signal-to-noise ratio), etc. This can be seen in Figure 6.4A.

(ii) Create meaningful subsets and only analyze that part of the data (Figure 6.4B). While the above automatic regression analysis provided an understanding on the entire test set, there could be a few more reasons of recognition failure. For instance, there might be a distorting factor on only a small subset of the data, which did not surface in the regression analysis because it was considering the entire dataset all together, e.g. a subset of audio files representing all females under age 10 might be having poorer recognition accuracy than the entire set, and could be a point to explore further.

(iii) See correlation trends of accuracy vs. specific metrics to identify reasons of failure. For instance, in Figure 6.5, the developer-user can make the following inferences:

   a. Different speakers have drastically different accuracy,
   b. Audio files that have low speaking rate (SPR) have poorer accuracy,
   c. Audio files with higher fundamental frequency (F0) have lower accuracy.

   After making these inferences, the developer can either drill down further at a speaker level to see what's wrong by selecting that part of data, or tackle issues with respect to SPR and F0 by applying specific optimization techniques. The exact technique to apply to counter each degrading variable is guided by our next module "Optimization Advisor."

(iv) Compare two (or more) subsets to understand what's different across those subsets. This would be particularly useful when comparing a subset giving high accuracy (>0.6), and a subset giving low accuracy (<0.6) to understand where exactly the two subsets differ. The comparison is based on the same seven acoustic features along with a few other meta-variables, e.g. speaker's age, gender, etc. Such a comparative analysis can lead to "developer-generated" hypothesis on why a recognizer may be failing.

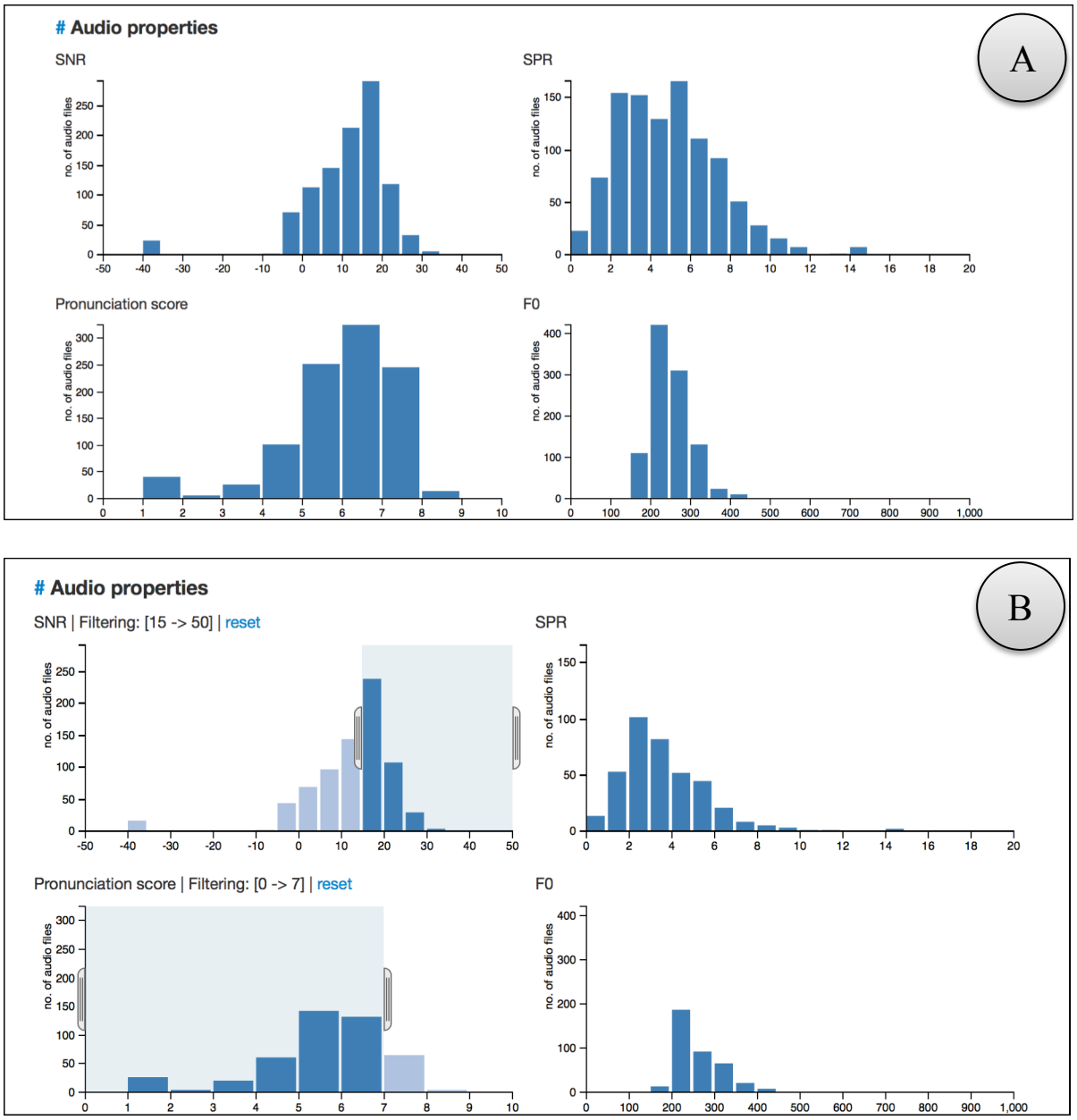


Figure 6.4: (A) Shows the original dataset, and its distributions on various parameters. (B) Shows a smaller subset where the audio files have poor pronunciation score (<7) and high SNR (>15).

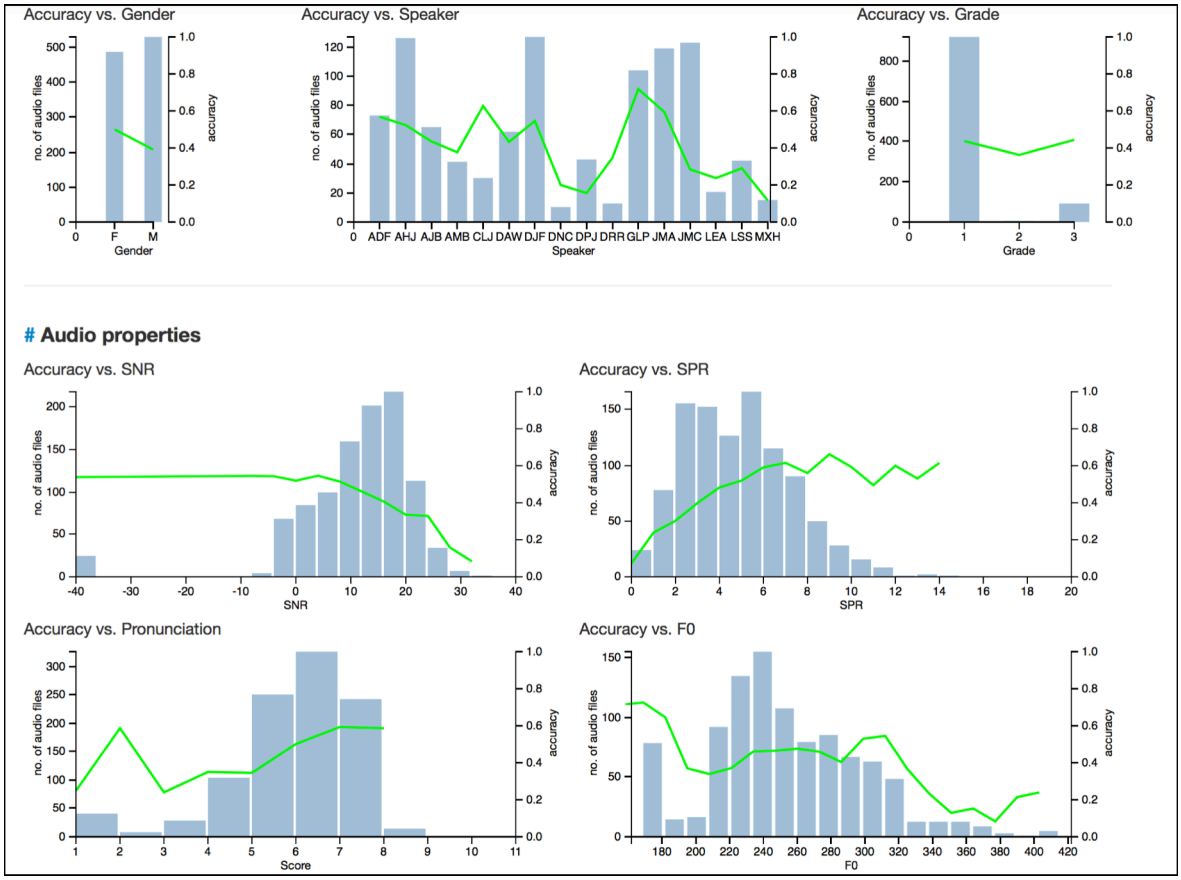


Figure 6.5: This figure shows correlations of accuracy (1 – word error rate) against several metrics, e.g. signal to noise ratio (SNR), speaking rate (SPR), etc.

This module was developed using several JavaScript API's that can rapidly process large datasets at client end, and display interactive graphs. In particular, we used angular.js [angularJS] for client-side routing and dividing content in different partial views. For generating bar graphs and line-graphs from CSV data, we used dc.js library [dc.js], which in turn relies on d3.js [d3.js] for rendering graphics in browser window. We also used crossfilter.js [crossfilter] to do fast map-reduce functions on data. The error analysis module also gives the ability to listen to the audio files right from the browser, and we use mediaelement.js to ensure that audio files run across different browsers and operating systems.

6.2.2.3. Out-of-Vocabulary and Substitution Analysis Tool

Another type of error analysis is to look at the recognizer's output and compare it with ideal output to understand patterns. For instance, the developer can compare the training vocabulary (set of words that the recognizer is capable of recognizing) with the words present in the testing transcript to identify the words that are not in the vocabulary – also known as out-of-vocabulary words. These words present a huge problem for speech recognition, and there's an entire thread

of research on it in the speech recognition community [Qin, 2013]. Another example is to look at the substituted words, and identify common patterns such as highly confused phonemes, or highly confused pronunciation. SToNE provides support for analyzing such errors.

6.2.3. Step 3: Optimization Advisor

The final but a key step in the speech recognizer development process is optimization. Once the developer has identified why the recognizer may not be working, they need to apply appropriate strategies to fix it. The role of the optimization advisor, as shown in Figure 6.4, is to advise the developer on what techniques to apply and how to fix the error. This module is based on the knowledge base developed in Chapter 5; however, it provides several supports to extend and improve the knowledge base.

Overall, the goal of the optimization advisor is to query the knowledge base and recommend appropriate techniques to the developer-users for their dataset. In doing so, it follows the ChAOTIC process, i.e. first it recommends techniques pertaining to "checking and cleaning" the dataset, then adaptation, then optimizations based on the most degrading factors, and so on. The developer-user can also query for another alternate techniques, and provide feedback to the optimization advisor, as to which of two worked better. Once the strategy has been recommended, the developer-user can refer to the step-by-step tutorial and the script for that specific technique in the virtual machine to obtain guidance on actually performing the optimization.

6.2.3.1. Refinement: Vote Up

For several problems, there are multiple solutions. This is because the knowledge base was a result of interviews with experts, and different experts offered a different solution for the same problem. To address this, we introduced a ranking system, where each solution had a rank, which was initially equal to the number of experts who had recommended a particular solution. We also provided a way to improve this rank based on actual results of what the developer-users found more useful in test runs through a "vote up" feature.

6.2.3.2. Extensibility: Experts Suggest a Solution

Since the field of speech recognition is still a hot area of research, the knowledge base needed to be updated every now and then. We provided a natural way of doing so, by enabling expert developers, e.g. researchers who have developed a new technique to insert corresponding solutions directly online. Once vetted by the central team for spam rules, this solution is added to the central database and available to the public. Initially, this solution starts with the lowest ranking, but over time, if it is a good solution, it gets voted up to become a higher ranked solution.

6.2.3.3. User Interface Recommendations

Once a recognizer has been developed with satisfactory performance, SToNE turns its focus on user interface (UI) changes. As demonstrated in Chapter 3 on "Voice Typing," these improvements are critical for two reasons: (i) UI changes can further lead to recognition accuracy improvements, which in turn, can improve usability. (ii) Human designers often may not be aware of the interplays between UI changes and the inner functioning of the recognizer, thereby making correct recommendations even more relevant. In its current implementation, SToNE supports implementation of the following two techniques, but given the rule-based design of the knowledge base, we envision that additional techniques can be easily supported without making any changes to the underlying framework.

Voice Typing: This technique was proposed and evaluated in Chapter 3. In this technique, users provide utterances in small phrases of 2-4 words, representing a thought-chunk, and then monitor (as well as, possibly correct) the output of the recognizer. We anticipate this technique to work for most cases when utterances comprise of full sentences or paragraphs (as in transcription of emails and text messages). It can also be useful in command-and-control interfaces where the command is spoken in a highly complex acoustic or language environment, and segmented audio can help improve recognition accuracy.

The advantages of this technique are to provide segmented audio to the recognizer, and stop any error propagation through real-time error correction, if possible. However, as discussed in Chapter 3, this technique cannot be universally used, e.g. for conversations where providing Voice Typing style segmented audio hampers the primary style of conversation, or in situations such as driving where a real "hands-free, eyes-free" interaction is needed and real-time error correction is not possible.

Recommending Semantically Similar Words: One of the common sources of error, as discussed in Section 6.1.1, is the substitution error. In command-and-control interfaces, words that sound alike are often confused for one another, primarily because there's no (or hardly any) context to disambiguate these words based on the language model. Even in large-vocabulary, continuous speech recognizers, many other common substitution errors occur. The goal of the optimization advisor is two-fold: (i) to identify such common confusions of error, and (ii) recommend alternates to the developer-user. For instance, in an isolated word, command-and-control interface, where the user has to say "Systems all ready" to achieve a specific task, and the common confusion is "System already", the optimization advisor could recommend a semantic equivalent to solve this problem, e.g. "System all set", if one is available after looking up for the synonyms on the web. Alternatively, the above information about the confusion

would enable the developer-user of the toolkit to think of other suitable alternatives, e.g. “All systems go”, “All systems set”, “All systems ready”, etc.

6.2.4. Relationship of SToNE with Deep Neural Network Systems

While deep neural network (DNN) systems have shown tremendous benefits in building speech recognition systems [Dahl *et al.*, 2012], SToNE is currently specifically geared towards aspects that DNNs cannot fix: out-of-vocabulary words, accent, speaking rate, etc. DNN training needs a properly setup and tuned GMM/HMM system, so one has to pick the correct optimization techniques and fine-tune a number of parameters, but an understanding of *what* and *why* to do is limited to experts. It is here that SToNE is particularly useful in. More so, DNNs work best with large amounts of data; however, developers & researchers are increasingly targeting newer user groups, languages or acoustic settings that lack existing data. SToNE specifically assists in such low-resource cases, and can enable better and more targeted data collection to iteratively improve the system even further. However, we envision that in future, researchers will extend SToNE to incorporate DNN techniques and can do so without fundamental changes to the overall architecture of current SToNE implementation.

7. EVALUATION OF STONE

Our earlier studies have shown great promise for the knowledge base, i.e. the ability to capture and formalize a speech expert's knowledge, and use it to analyze and predict optimization techniques for the benefit of novices in speech recognition. In this chapter, we focus on the summative evaluation of SToNE towards achieving research goal 3 (RG3). In particular, we describe results from a user study where non-experts are asked to use the toolkit and build a speech recognizer for a challenging use case, i.e. children's conversational speech. We benchmark their results against our attempt, i.e. an expert team's attempt at building a speech recognizer for the same use case.

We have situated our evaluation in the context of building a large vocabulary continuous speech recognizer (LVCSR) for children. This is for several reasons. First, building a recognizer for children is a challenging task in and of itself because their voice, as such, presents many challenges not present in adult's speech, e.g. hesitations, repetitions, variable vocal tract length, etc. Second, there are no off-the-shelf recognizers for children readily available, from which a new one can be bootstrapped for development easily, and hence, researchers and developers have to build the recognizer ground up. Finally, it was important to pick a task that was not widely published so as to evaluate the toolkit with a non-trivial task, and also control for any knowledge that developer-users could gain from simply searching the web.

7.1. Building an LVCSR for Children: Benchmarking by Experts

First, we describe our own efforts at building an LVCSR for children in grades 4-8 using the conventional approach. This phase was important for two reasons: (A) as a feasibility check to validate that the task was achievable by a team of experts at the least, and (B) to serve as a benchmark when comparing the improvements achieved by using SToNE later in this paper. For this section, our contribution is not in terms of proposing a new technique, but in reporting the results for a previously unreported task, i.e. an LVCSR for children.

The experts who were involved in this task were all senior PhD or Masters students at Language Technologies Institute of Carnegie Mellon University. They had at least 2 years of prior experience building speech recognizers, and had specific expertise in at least one of the speech recognition areas: acoustic modeling, language modeling, or dictionary development. However, none of them had previously worked with children's dataset, which made the choice of

children's LVCSR a perfect choice as it avoided any confounding variable of a previous experience in the benchmark.

7.1.1. Related Work: Speech Recognition for Children

While children's speech recognition has been studied a lot, no recognizer for children's conversational speech has been built. For instance [Steidl *et al.*, 2003] focus on using an adult recognizer and adapting it for children, whereas [Giuliani *et al.*, 2003] train a children's recognizer using structured, read speech from children. Other work has focused on more specific use cases: for instance, games developed at UC Berkeley [Tewari *et al.*, 2010] focus on improving pronunciation skills for children by providing interactive feedback for poorly articulated or mispronounced words. CMU's reading tutor LISTEN [Mills-Tettey *et al.*, 2009] helps children read pre-determined sentences, and learn new words in the process. It presents a sentence at a time from a storybook, asks the child to read it aloud, and provides feedback when the reader is stuck. Similarly, another research group at CMU built language learning mobile games for children [Kumar Reddy *et al.*, 2012] that encourages them to say vocabulary words aloud, and in the process helps them learn new words. A primary difference between our work and previous work by others is that while others have used speech recognition to solve specific issues for children, e.g. improving pronunciation, detecting misreading, teaching new words etc., we focus on a much more challenging problem, i.e. the task of recognizing open-ended speech. Although researchers have collected datasets to address this problem [Shobaki *et al.*, 2007], to our knowledge, there's no published work that reports on the attempts of building an LVCSR for children.

7.1.2. Dataset C: CSLU Kids Corpus

We used a published dataset from Linguistic Data Consortium on children's speech [Shobaki *et al.*, 2007], which comprises of audio files from 1101 children between kindergarten and grade 10, equally split across gender and grades. The audio files, in total, contain about 40 hours of speech. There are two parts to the above dataset. The first part has isolated words read by children – each child read 60 items from a total list of 319 phonetically balanced words or sentences. The second part contained utterances of spontaneous speech from each child. These utterances begin with a recitation of the alphabet and then contain a monologue of about one minute in duration. The monologue was in response to data collector's questions such as "tell me about your favorite movie." Corresponding word-level transcriptions are also included in this dataset. As we are focusing on spontaneous speech, we ignored the first part, which was as such, very small in terms of total speech. This dataset is one of the very few published datasets that contain substantial amount of children's speech, and hence was a natural choice for us. For the purposes of the development, we split this dataset into data from 999 speakers as the train set, and 102 speakers for the test set. The test set, however, only contains data from grades 4-8,

which was our target group. Although this dataset has been around since 2000, there is no substantive work done on it because of its challenging nature. The only published work citing this dataset is that of [Shobaki *et al.*, 2000], but even here, they have only focused on isolated speech subset of the data.

7.1.3. Methodology and Results

We followed the six-step methodology as describe above to iteratively build and improve the speech recognizer. To build the recognizer, we used the open-source Kaldi speech recognition toolkit [kaldi], primarily because it has support for many of the state-of-the-art optimization techniques, such as discriminative training and neural networks. The primary evaluation metric was word error rate, and our results are summarized in Figure 7.1.

We started by checking if an existing LVCSR recognizer can be bootstrapped for our task. To check its feasibility, we evaluated our test set with multiple available LVCSRs, built on adult conversational speech using the Switchboard dataset [Graff *et al.*, 2004], or Wall Street Journal dataset [Paul *et al.*, 1992]. The word error rates in these cases were terribly poor, i.e. well above 90%, and suggested that we should build the baseline ground up from the provided training set.

While building the baseline ground up also requires successfully completing a number of tasks, existing toolkits such as Kaldi provide sample scripts that can simplify the early setup. It still, however, leaves a number of parameters to be edited by the developer based on their dataset. Setting these parameter values properly requires experience in using Kaldi. For instance, for a speech recognizer to be built, the first step is alignments where the recognizer attempts to align a given audio file with its corresponding transcript. This alignment process can be very time intensive, and researchers typically optimize it by pruning a large search space into a smaller

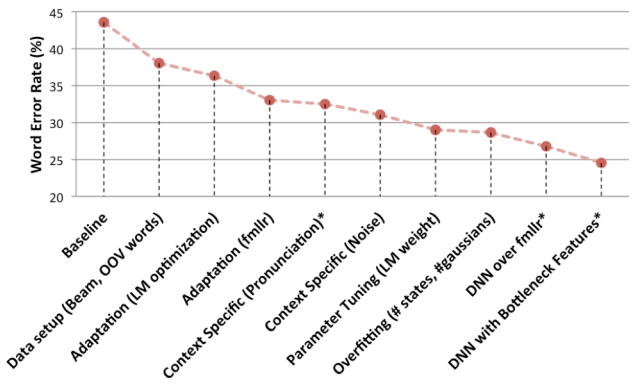


Figure 7.1: Word Error Rate improvements over successive development of children's LVCSR by a team of experts. The current implementation of SToNE supports development of all techniques listed above, except the ones marked with a (*).

search space. Two parameters help in this pruning "beam size" and "retry beam size." A larger value for these parameters implies longer time to find time alignments, but also implies that longer audio files (>60 seconds) can now be properly aligned. Conversely, attempting to align a long audio file with the default values of these parameters in the sample scripts leads to misalignments. In other words, there is a tradeoff between alignment accuracy vs. computation time, and the exact value should be dependent on the average length of the audio files. Our audio files were about 90 seconds long. We set "beam size" value to 100, and "retry beam size" to 300. Our baseline acoustic model had 1800 states and 9000 Gaussians, and the language model weight was 12. The WER for this baseline system was 44.57%.

Once the baseline was properly configured, we next identified out-of-vocabulary (OOV) words. These words were inherently impossible to be recognized as they were not present in the dictionary, and so the recognizer was attempting to find the next best match for them. 24.36% of words in the training dataset, and 11.24% in the testing set were out-of-vocabulary. We generated pronunciations for the OOVs in the training dataset using the sequitur g2p tool [Bisani *et al.*, 2008], and added those to our dictionary. We also retrained the language model to include the unigram probabilities for the OOVs. The test set OOV after this setup was 3.67%. The WER of our system after the data setup step was 37%.

In the general adaptations category, we optimized our language model using Kneser-Ney smoothing [Knesner *et al.*, 1995], and our acoustic model using feature-space maximum likelihood linear regression (fmllr). These are widely used optimization techniques, but need to be selected from a host of similar options depending on the context of use and the requirements of the recognizer. As mentioned before in Chapter 5 that if selected correctly, general adaptations almost always tend to improve the recognition accuracy. In other words, it requires an understanding of *goals* (speaker-dependent vs. speaker-independent), and *knowledge* of the techniques (Kneser-Ney, fmllr, mllr, map, etc.) to be able to select the correct technique and improve the recognizer's performance. The WER of our system after this step was 33%.

In the context- and user-specific adaptations category, we first identified two major causes of error: children's pronunciation and noise. Accordingly, we updated the dictionary with alternate pronunciations and applied a normalization technique called Cepstral Mean and Variance Normalization (CMVN) [Chen *et al.*, 2002] to account for noise and channel variations in the training dataset. Although some experts prefer to apply normalization techniques early on, i.e. before the general adaptation techniques, from an iterative development perspective, it does not matter when you apply it as long as you identify it as an issue at some point in the development process and correct for it. The WER of our system after this step was 31.1%.

Thereafter, we also tuned a few parameters, e.g. language model weight, and adjusted for overtraining by changing the number of Gaussians. The language model weight was determined based on several decoding runs with language model weight ranging between 10 and 20. We picked 14 as it gave us best results on the development set. Our acoustic model after this step had 5000 states and 60000 gaussians. We had arrived at this number after a few attempts as well, with a guiding principle that each gaussian should roughly account for 50-100 milliseconds of audio in the training dataset. Such parameter optimizations are often difficult to identify without having some previous experience. Experts who have worked on building recognizer for years have usually developed a few heuristics to arrive at a reasonable range of values, and then try out a number of combinations within this reduced range. The WER after these steps was 28.65%.

After this point, we developed a deep neural network (DNN) model, which are built on top of GMM/HMM models developed above. We used the Kaldi + PDNN toolkit [Miao, 2014] for this, which is a wrapper built on top of Kaldi [kaldi] to easily build DNN systems. The rough idea behind this wrapper tool is to build the initial GMM models with Kaldi, train DNNs with PDNN [Miao, 2014] and finally load the DNN models back to Kaldi for further decoding or system building. DNN systems are the state-of-the-art speech recognition models [Dahl *et al.*, 2012], and experts have very recently begun to adopt DNN training in their development routines. The WER after this step was 24.47%.

It is important to note that DNNs – the state-of-the-art speech recognition technology – still relies on properly fine-tuned and optimized GMM/HMM model. For instance, we experimented with simply using DNN techniques after building a baseline model (WER of 44.57%). The resulting recognizer only demonstrated marginal improvements at a WER of 41.27%, i.e. an absolute improvement of 3.3%, and a relative improvement of 7.4%. On the contrary, the DNNs that were developed over an optimized and fine-tuned GMM/HMM model had an absolute improvement of 4.18%, and a relative improvement of 14.6% - almost twice as much. These results have two implications: (i) DNNs, in general, are useful to improve WER, and (ii) when applied over a properly optimized and fine-tuned GMM/HMM system, they demonstrate far better gains than over simple baselines.

7.2. Evaluation of Toolkit with Non-Speech Experts

In order to explore the efficacy of SToNE in comparison to traditional methods of building a speech recognizer, we ran a controlled experiment. In this experiment, participants engaged in a speech recognizer development task. We also wanted to assess the level of utility of SToNE for developers with varying levels of skill in speech: (1) Non-experts: no practical experience and no theoretical background in speech recognition, *vs.* (2) Semi-experts: no practical experience in

building a recognizer, but had attended a speech course, which made them familiar with the speech terminology and the basics of speech recognition. Because practical experience is vastly different from theoretical knowledge, we expect that our toolkit will also be useful for semi-experts as well. Additionally, we restricted the development timeline to 10 days as the purpose of SToNE is to make iterative testing of speech prototypes possible by rapid development of accurate, working recognizers, and hence evaluation under limited time is of value.

7.2.1. Participants

We recruited 12 paid participants via mailing lists and a research participation pool. Subjects were graduate and undergraduate students from a computer science background, i.e. they were familiar with basic computer science and programming fundamentals, e.g. how to run a script, or basics of Linux. All participants were between the ages of 20 and 26. Of these 12, six participants had taken a graduate level course on speech recognition and were therefore familiar with the basic concepts of a speech recognizer. As part of the class, they had also worked on small assignments related to speech recognizer development, but never developed an end-to-end working speech recognition system. We call these six participants *semi-experts*. The remaining six participants had no prior understanding of speech recognition, and are referred to as *non-experts* below. The participants were drawn from the same pool of students as the experts in Section 7.1, but the experts had far more practical experience in building systems than both the non-experts and the semi-experts.

7.2.2. Experimental Task and Setup

Every participant was asked to develop a large vocabulary continuous speech recognizer (acoustic model, language model, dictionary) for children in grades 4-8. As mentioned earlier, the choice of task reflected the challenges that a developer may encounter while building a recognizer for any other user group, acoustic or language situation.

For this task, we gave each participant a training set of audio files (with associated transcripts) from 999 speakers representing users in kindergarten to grade 10, and a test set of 102 speakers from grades 4-8. In other words, the initial setup was same to what we had used in building the recognizer on our own.

7.2.3. Experimental Design and Data Collection

We conducted a 2 x 2 between-subjects experiment with two independent variables: *speech toolkit* (SToNE-built-on-top-of-Kaldi vs. Kaldi alone) and *level of expertise* (non-experts vs. semi-experts). Below we refer to the "experiment" condition as the users who used SToNE-built-on-top-of-Kaldi, and "control" condition as the users who only used Kaldi. In total, 6 participants (3 non-experts, 3 semi-experts) were assigned to the experiment condition, and 6 (3 non-expert, 3 semi-experts) were in the control condition. For the experiment condition, we

provided the users access to our full toolkit as described in Chapter 6 and also in Section 7.2.2, whereas in the control condition, we only gave users access to the virtual machines without any of the sub-modules from SToNE, but with pre-installed and ready-to-go environment for Kaldi and other (default) example scripts. Our hypotheses were:

> *H1: SToNE will assist both semi-expert and non-expert developers achieve* ***better accuracy*** *with* ***higher task success rates****.*
>
> *H2: With the help of SToNE, non-expert and semi-expert developers will be able to achieve* ***accuracy rates similar*** *to that of experts.*

In terms of data collection, we collected the following: before the experiment, we collected basic information about each participant, e.g. age, years of experience with building computer science applications or programming, years of experience (if any) with speech systems, speech courses taken, etc. During the experiment, we collected logs of all the actions of the developer-user, to measure the word error rates achieved, and task success rates. The study with each participant took place over 10 days, and they could complete the task over multiple sittings, at home as per their convenience. We also met with each participant for 15 minutes during the study to check their progress and resolve any issues. At the end of the experiment, we conducted a post-hoc interview with the participants to understand areas where the toolkit violated novice-developer expectations, and how it could be further improved to incorporate the variable skills sets of the novice speech application developers.

For our analysis for both hypotheses, we controlled for prior experience (number of years working with speech recognition), time spent on the error analysis, and time spent on actual optimizations (in virtual machine). Time information was obtained from the logs. To completely evaluate both the hypotheses on a held-out test dataset, we required a working recognizer from the participants; however, in cases, when none was obtained due to incomplete task completion, we looked at the logs in the virtual machines to obtain an understanding for where the participant stopped and why the participant was unable to complete the task.

7.2.4. Results

7.2.4.1. Word Error Rates

In order to compare the accuracy achieved by each participant, we calculated Word Error Rate (WER), as discussed in Section 5.3. The word error rate that each participant achieved is shown in Figure 7.2. We omitted the graph for the non-experts (who did not use SToNE), as they did not proceed beyond selecting an off-the-shelf recognizer, and evaluating it with the given test set. The WER was well above 90%, identical to the first attempt made by experts. The three

semi-experts who did not use SToNE were able to correctly setup a baseline model using the given training dataset, and then apply a few optimization techniques. The final WER they reported were 40.5%, 40.7%, and 41.2% respectively.

For the participants who used SToNE, all non-expert and semi-expert users were able to build a baseline model, and then apply a number of optimization techniques. The three non-experts reported a final WER of 29.4%, 29.8%, and 30.8%, while the three semi-experts reported a final WER of 29.4%, 29.8%, and 30.1. On average, this is a 27.3% relative improvement in WER over the semi-experts who did not use SToNE. The improvements achieved by SToNE users over users who did not use SToNE were statistically significant, $p=0.02$. More so, the improvements achieved by non-expert SToNE users (the lowest skilled SToNE users) alone over semi-expert users who did not use SToNE (the highest skilled non-SToNE users) was also statistically significant, $p<0.01$.

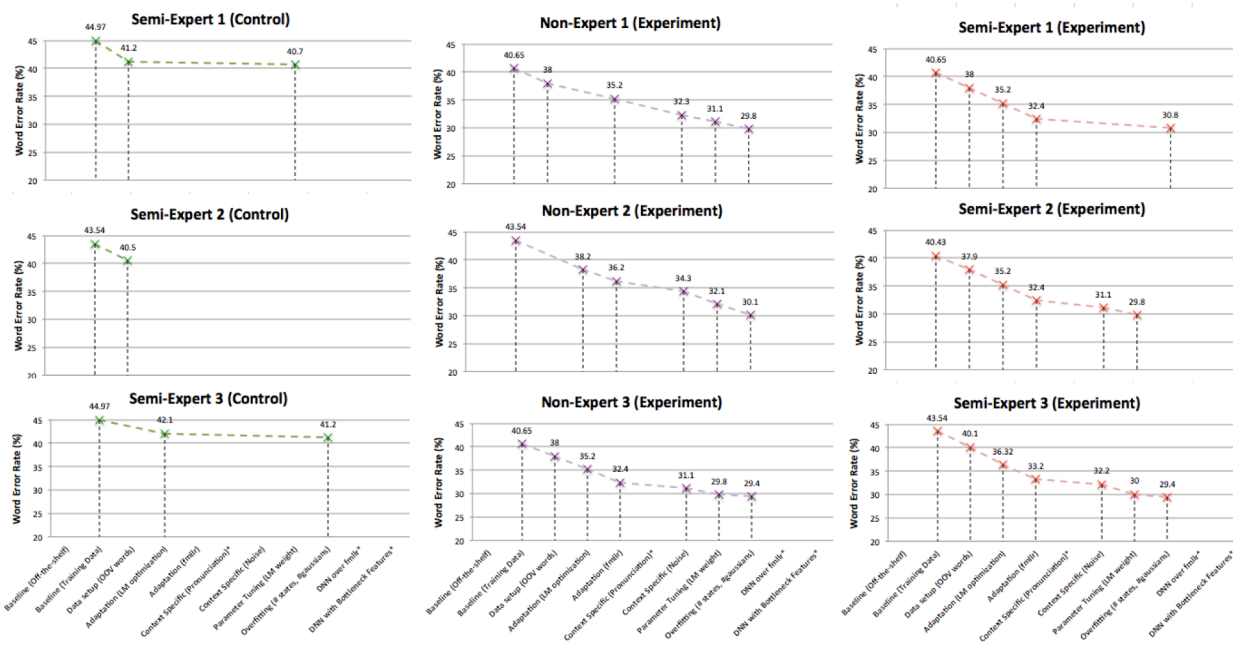


Figure 7.2: The *experiment* condition referred here are the participants who used SToNE, and the *control* condition here are those that did not use SToNE. The word error rate improvements by semi-experts in the control condition, and non-experts and semi-experts in the experimental condition. Note we did not plot the word error rate improvements for non-experts in the control condition as they did not get beyond a baseline in the development process.

7.2.4.2. Task Success Rate

To contrast the number of attempts that participants made, and understand their success rate, we measured the number of times that people attempted a new configuration setup, i.e. a new optimization or a new parameter setting, and were successful or unsuccessful. As shown in Figure 7.3, in the control condition, both the non-experts and semi-experts tried a number of

configurations, but their success rate was lower at 25% for non-experts, and 40% for semi-experts. In the experiment condition, however, the success rates for both non-experts (71.4%) and semi-experts (83.3%) were higher. Moreover, the average number of attempts in the experimental condition was also higher – 6.5 total attempts vs. 4.5 total attempts. The two-factor analysis of variance showed no significant main effect for the *expertise* factor, $F(1,8) = 0$, $p > 0.10$; a significant main effect for the *speech toolkit* factor, $F(1,8) = 16$, $p < 0.01$, and no interaction between expertise and speech toolkit factors, $F(1,8) = 4$, $p = 0.08$.

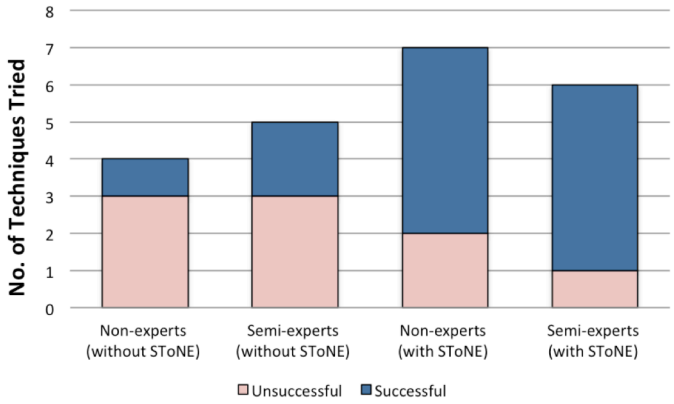


Figure 7.3: The average number of techniques attempted by participants. "Successful" implies techniques that helped in reducing the word error rate. "Unsuccessful" techniques did not make a difference or increased the WER.

7.2.4.3. Time Spent on Task

In order to further understand whether the differences in "success rates" were because of added time spent with the interface, or a factor of guidance from the toolkit that enabled the developers, we measured the time that they spent working on the interface. Since the toolkit's front-end interface was hosted on the web (in the experiment condition), and on the virtual machine (in the control condition), we measured the time spent through logs, which were continuously recorded for every action in the background.

In fact, we recorded the following actions: every time a user opened the interface, every time they made an action (selection, starting a training, stopping a training, parameter change, etc.) Similarly, in the virtual machine, we noted the time the participants logged in and logged out, and also the time they spent making changes to files or directories. To avoid counting dormant times, we threshold a session at 15 minutes, implying that if the user was inactive for more than 15 minutes, we did not count that duration in the time spent on working on the interface.

Figure 7.4 shows the average time spent by both non-expert and semi-experts in each condition. The two-factor analysis of variance showed no significant main effect for the *expertise* factor, $F(1,8) = .70$, $p = 0.42$; no significant main effect for the *speech toolkit* factor, $F(1,8) = 3.44$, $p =$

0.10, and also no interaction between expertise and speech toolkit factors, F(1,8) = 2.95, p = *0.12*. This implies that the time spent on the interface is more or less same for all conditions. This serves as a sanity check for our results that the gains achieved are not a function of the total time spent with the interface, but more so of the time spent correctly, which was enabled by the guidance of SToNE in the experimental condition.

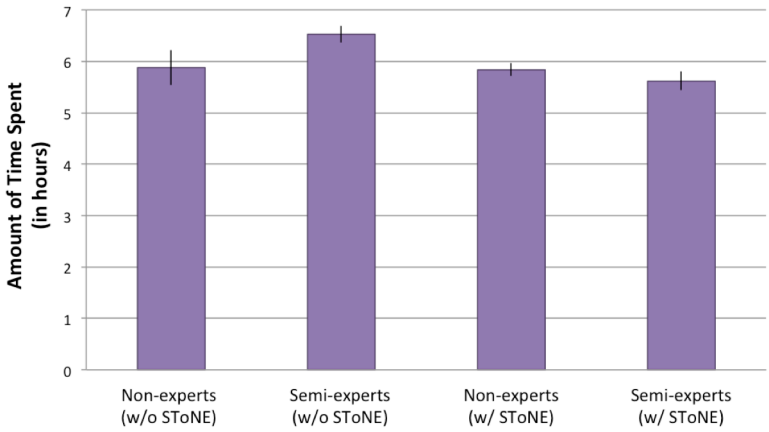


Figure 7.4: The average time spent by participants working on the children's LVCSR task.

7.2.4.4. Open-Ended Comments

With respect to open-ended comments, participants seemed to appreciate the optimization advisor aspect of the toolkit. As one non-expert participant put it:

> *"The toolkit was very helpful because it gave me a simple and abstracted way of developing an ASR system. It also informed about various factors that can bring down the WER and corresponding solutions."*

The process of building the recognizer was also informative and educative for a few participants. A non-expert participant highlighted that:

> *"I learnt that building an ASR system is an iterative process. Many parameters and solutions are dataset specific. A parameter value, which worked for children's speech data may not work for any other training corpus."*

Another participant highlighted the iterative nature of the development process:

> "*I have learned that it is a long process to develop a good recognizer. It's good to make changes/improvements in small increments and see how tweaking single options affect your system instead of using a shotgun approach and changing many things at once.*"

This was a tremendous gain in understanding given that the non-experts had no prior knowledge of how speech recognizers worked internally, but left with an open understanding that there are many parameters that need to be optimized, and the exact optimization is dependent on the dataset or the use-case. There is no one-size-fits-all solution. In other words, it may help thwart concerns of non-expert developers as in [Laput *et al.*, 2013] that if an off-the-shelf recognizer is not working, they have no control over what to do or how to improve it.

At the same time, participants expressed difficulty in identifying correct parameter values:

> "*Once I knew that a parameter value could be improved, it was difficult to zero in on a value. Even after several tries I was not sure whether I have reached the best possible value; although that might be impossible to find, but some understanding on how close we are to the optimum value might be helpful.*"

This was true. While we provided guidance on what parameter to change, and which direction to change it, there was no guidance on how much to change it by. As each training run took almost a day, participants tried a few combinations of values only before moving on to other types of optimizations.

7.3. Discussion and Future Work

Our studies demonstrate that SToNE can help non-experts and semi-experts alike in building and improving the accuracy of a speech recognizer. We offer a number of concrete takeaways from our results:

As expected, in the control condition, semi-experts performed much better than non-experts: semi-experts were able to create systems with ~40% WER, while non-experts failed entirely. This result was expected. Semi-experts – developers who have attended a theoretical speech course – have some understanding of why recognizers may work or fail. This understanding was useful in setting up a proper baseline. However, due to their lack of experience in building working recognizers (or working with any toolkit), they were unable to find optimization techniques that could improve the recognition accuracy any further.

With the help of SToNE, all semi-experts and non-experts performed almost as good as experts, creating systems with ~30% WER. This result suggests that semi-experts and non-experts benefitted similarly from the guidance provided by SToNE. Although an application with a WER of ~30% does not seem to be directly usable by end users, we believe that this is a strong result because of two reasons: (i) participants were able to achieve WER similar to experts, implying that they would likely succeed in building usable recognizers when experts are also

able to. (ii) Having a reasonable system, i.e. with WER ~30% enables the developer-users to deploy a working system to collect more data from end users (as opposed to conducting Wizard-of-Oz studies), and improve the system over time.

Even though the final 30% WER achieved by non-experts may seem high on children's LVCSR task, they are now able to create useful speech systems. The reasons are many: (i) as mentioned above, a 30% (vs. 90%) WER system can be used to field a "semi-working" system to collect more and better data, which can in turn further improve WER. (ii) Moreover, there is no clear-cut answer to the question "how good is good enough for speech recognition?" For instance, 25% has been shown to be adequate for transcription tasks [Munteanu *et al.*, 2006]. Similarly, despite high WER in about 50% range, spoken dialog systems have triangulated other sources of information to achieve good task success rates [Lopes *et al.*, 2013]. In other words, overall success depends on more than just the speech recognition accuracy alone. (iii) Finally, and more importantly, the WER reductions achieved by non-experts were similar to that of experts on building a children's LVCSR. For other, simpler tasks where experts may achieve lower WER, we imagine non-experts – with the help of SToNE – will also achieve lower WER that result in a deployable system directly.

Non-experts (in the experiment condition) achieved better performance than semi-experts (in the control condition). This result shows that SToNE can benefit any developer-user who does not have a practical experience in building speech systems, to an extent even greater than what semi-experts can achieve without SToNE.

Participants using SToNE generally had a higher task success rate than those not using it. This result showed that developer-users using SToNE were able to try many different configurations, and guided by the toolkit, a larger proportion of their attempts improved performance.

Participants in the experiment condition used different combination of techniques to achieve near similar performance. This result suggests that although SToNE enables the developers in understanding what the potential problems may be, and guides them how to solve those, it does not mandate any specific action. This was by design. While the machine can always try a brute-force strategy to find the best combination of techniques, this was not done for two reasons: first, this will require massive computing power. For instance, with just 10 techniques, the total number of combinations possible is $2^{10} = 1024$. With each combination requiring almost an entire day to run on 8 cores, this simple setup can take almost three years to complete. Worse, the time will increase exponentially with the total number of techniques and the training data size. Second, often achieving optimum performance may require some changes to the user

interface or the interaction style, which are impossible to be validated without a human developer in the loop.

Participants spent multiple attempts tuning parameter values. As mentioned earlier, in a speech system, there are multiple knobs (parameters) to optimize. In our current implementation of SToNE, we were able to guide the developers with respect to the *direction* in which to move their parameter values to further improve the WER. However, this strategy still required the developers to manually try out a few values before reaching an optimum. Since each try can take significant time to complete, such a discovery method can be nonetheless slow.

In future, we intend to focus on automatically determining the optimum *exact* values for such parameters. In fact, researchers have already begun to explore such a research direction, and more formally, it's known as black-box optimization [Watanabe *et al.*, 2014]. Using this method, we can identify *exact* value from a million combinations automatically without performing a brute-force search.

Even though we have situated this task in the context of building a large-vocabulary continuous speech recognizer, we believe SToNE could eventually be generalized to help in the development of other types of speech interfaces, e.g. with different sizes of *vocabulary* (tiny, small, medium, large), different types of *conversation* (isolated, command-and-control, continuous), and different *amounts of training data* (low, medium resource, or large). Although our prototype does not support all the state-of-the-art techniques – a "moving target" as such, we believe that its current implementation as a rule-based knowledge base can be easily extended to support other adaptation techniques without requiring fundamental changes to the overall infrastructure.

8. DISCUSSION & CONCLUSIONS

This thesis has investigated how to support development of speech user interfaces by those who don't have speech recognition expertise, but are interested in using functional speech recognizers in their research or application development. As we conclude, we summarize the contributions in this thesis, and discuss avenues for future research.

8.1. Summary of Contributions

We summarize our contributions in terms of three high-level stages: (i) *identification* of the speech user interface development problem, (ii) *design* and *development* of tools to lower the bar of speech user interface development, and (iii) *evaluation* of the above tools with non-expert and semi-expert users.

8.1.1. Identification of Speech User Interface Development Problem

In **Chapter 2**, we discuss the development of a speech recognizer for children in rural India. This was a significant speech recognition problem as the users were non-native speakers of English, had significant variations in voice as is expected of children in grades 4-5, and the deployment of the application was in a noisy background. Through successful development and deployment, we discuss the challenges involved in building an accurate SUI, and the lessons we can learn to facilitate the development process for other non-expert developers.

In **Chapter 3**, we discuss how once a reasonable speech recognizer has been developed, we can leverage that to make changes in the interaction style of the speech application. We showed that these changes can in turn improve recognition accuracy, and further the usability of a speech application. In other words, we demonstrated that once a recognizer has been developed with sufficient accuracy, changes to the interaction style are enabled and those could lead to further gains in recognition accuracy. To demonstrate this, we design and develop a novel interaction technique called "Voice Typing" that reduces the recognition error rate in modern day dictation interfaces by a relative factor of 27%.

In **Chapter 4**, we bring together the lessons from both the above case studies to discuss four design myths. We further debunk those myths with empirical evidence, and argue why reasons for why speech recognition and HCI communities should be tightly coupled for the development of a usable and useful speech user interface.

8.1.2. Design and Development of Tools to support SUI Development

In **Chapter 5**, we discuss the design and evaluation of an expert knowledge base to facilitate development of speech recognizers. To do so, we conduct contextual interviews with speech experts to understand the process, knowledge, and intuition that they use while building speech systems. We then formalize this knowledge in a rule-based knowledge base, and evaluate its predictive power with two English language datasets that have significant challenges for speech recognizer development such as children's speech, background noise, or variable speaking rate. Through the formalization of this knowledge base, we provide a recommendation and analysis engine specific for speech recognizer development.

In **Chapter 6**, we further develop a set of speech analysis and recommendation tools for non-speech experts that can assist non-speech experts by automatically and semi-automatically analyzing reasons of recognition failure, and then recommending optimization techniques for tackling those issues. These tools include a: (i) feature extraction module for pronunciation scoring, signal-to-noise ratio extraction, speaking rate, etc. that quantifies specific signals for understanding the acoustic context of a given dataset, (ii) an analysis module that identifies the significant degrading factors for the speech recognizer and generates appropriate visualizations for further analysis, (iii) a visualizations and comparison module that helps analyze reasons of failure in subsets of data, and (iv) a recommender module that guides non-expert developers on specific steps to make the recognizer work better.

8.1.3. Evaluation of SToNE

In **Chapter 7**, we evaluate SToNE with both non-experts and semi-experts. We report on a couple of key findings: (i) in the absence of SToNE, semi-experts outperform non-experts, (ii) with the guidance of SToNE, both non-experts and semi-experts report significant accuracy improvements on their speech recognition task (about 30% relative reduction), and their accuracy was close to that achieved by experts on the same task, and (iii) with the help of SToNE, the final accuracy achieved by non-experts and semi-experts is not significantly different. In other words, non-experts and semi-experts were able to build useful systems with the help of SToNE, which can then be fielded to collect more and better data, to iteratively improve the system further.

8.2. Limitations

A key component for an ASR system is a pronunciation dictionary, which maps each word to its phoneme sequence. Each language has its own phoneme set, and even within each language, different dialects may have different phoneme sequences for the same word. More so, every

user group may have a different way of articulating a word aloud. These differences lead to difficulties in standardizing pronunciation dictionaries, and one must be developed for every new ASR. Although there are g2p tools [Bisani *et al.*, 2008] to automatically generate pronunciations, these require a training dataset to learn letter-to-sound rules, and then apply them on unseen words. However, for resource-scarce languages or dialects where ASR development is fairly new, pronunciation dictionary development is still more suited for linguistic experts than a machine [Davel *et al.*, 2009]. This is because properly generating (alternate) pronunciations of a word is accurately possible only with years of linguistic knowledge. Linguistics use their own error analysis methods to ensure consistency [Martirosian *et al.*, 2007], which can certainly be incorporated in SToNE in the future. In its current implementation, SToNE has a g2p tool to automatically add pronunciations for unseen words, but it does not have supports for building a pronunciation dictionary ground up.

Similarly, text normalization is another area where experts do a much better job than any automated script. While developers can write general-purpose scripts to normalize transcripts during the data setup phase, i.e. ensure that everything is consistent (no punctuations, lowercase vs. uppercase, spelling errors, etc.), each transcript follows its own set of conventions, which are hard to incorporate in a general-purpose script. SToNE so far only support the general-purpose normalization tasks, and understanding specific transcript conventions is still left to the developer. Sometimes, this task is best suited for data experts who have years of experiential knowledge at cleansing and filtering large volumes of data.

Another limitation of SToNE, in its current design, is the lack of support for deep neural network training. Deep neural nets are at the forefront of speech recognition research, and have demonstrated to outperform typical GMM/HMM models, both in our task on children's LVCSR in section 7.1.3 and for other tasks [Dahl *et al.*, 2012]. However, they require heavy computational power, and typically run on graphical processing units (GPUs), as opposed to standard central processing units (CPUs). Our current implementation setup of SToNE hasn't incorporated GPU-based training, but its design can support extension to such a task in future.

8.3. Opportunities for Future Research

Through this dissertation work, we have begun to formalize speech recognition experts' knowledge, and developed a number of tools to support the development of speech recognizers by non-expert developers. Now towards the end, we enumerate a number of research and development opportunities, which will benefit from a more thorough investigation in future.

8.3.1. Black Box Optimization for Speech Recognition

As mentioned in Chapter 1 and Chapter 7, at a high level, there are two primary aspects while developing an accurate speech recognizer: identifying correct optimization techniques, and finding optimum parameter values. SToNE was good in recommending appropriate optimization techniques and also was able to guide the developers the *direction* in which to change the value of parameters. However, the exact value of the parameter was still left to the developer, and while the toolkit currently provided some guidance to the developers on appropriate ballpark values for the parameters, finding optimum values still required a few attempts. Because each attempt in speech recognition can take hours (if not days) to complete, this method can be both time intensive and frustrating. This was also indicated in the qualitative feedback from a few participants. We propose that future researchers attempt to develop techniques to find optimum values for these parameters automatically, similar to black box optimization [Watanabe *et al.*, 2014]. A few research questions to consider:

- To what extent can we automatically determine the optimum values of parameters in a speech recognizer? Which parameters are best suited for black box optimization discovery, and which are not? What additional information may be needed for such parameters?
- Once we have the optimum parameters for a few settings, how can we integrate the meta knowledge from these settings to discover optimum parameters for other settings more easily?

8.3.2. Neural Network Training

In Chapter 7, we showed that experts benefited from applying advanced techniques such as neural network training to achieve significant gains towards the end. The improvement achieved in word error rate was from 28.65% to 24.57%, a 14% relative improvement. While this technique can be highly rewarding and has proven to be beneficial in many development situations recently [Dahl *et al.*, 2012], it is also highly challenging. First, it introduces another set of parameters, such as number of layers, learning rate, number of nodes per layer, etc. that need to be appropriately adjusted depending on the task. Second, to train a neural network system, one needs highly powerful computers. Even then, it takes three to four times longer to complete the training. In the future, researchers can explore how to speed up this process, and integrate with SToNE's existing infrastructure. Possible directions could be Amazon's Elastic Compute Cloud (EC2) service, which provides a scalable way of hosting computing on central servers at Amazon for a small fee.

8.3.3. Supporting Discovery of Development Requirements (for Designers)

Another informal feedback that we heard at ACM CHI's workshop on Designing Speech and Language Technology [Munteanu *et al.*, 2014] where we present the toolkit was that designers

of speech user interfaces are often perplexed as to "what information they need to collect before passing it on the developer?" While this problem is one of requirements gathering (an assumption we made while thinking about the toolkit), we can imagine a framework for designers in the form a set of questions or probes that they can use to think about a speech technology design and development. Future researchers in speech user interface technology design can hopefully work towards such a universal framework for designers.

8.3.4. Active Learning for Selective Data Transcription (for Developers)

In this thesis, we have focused on developing a functional speech system, often with low resources. Once a "working" or "semi-working" system has been developed, we have also discussed that such a system be used for additional data collection to iteratively improve the WER. While additional data collection is useful, the associated transcription can be costly. At this point, developers need to get selective and identify parts of data that will benefit the system most from transcription. How do you automatically identify such subsets is a research question that might benefit from future investigation?

There are already some threads of research focusing on this aspect, and termed under "active learning for speech recognition." [Riccardi *et al.*, 2005] We propose that such research can also benefit from integration with our automatic or semi-automatic analysis module that identifies the most significant degrading factors and then intelligently selecting subsets based on this information.

8.3.5. Applicability to Other Languages

Another thread of interesting research is the extension of our work in developing expert knowledge base for other languages. To our understanding, the rules and techniques in our knowledge base are primarily generic, but a few language specific issues might have to be incorporated to make it work for other languages. What techniques are transferable and universal across all languages? Are there techniques that work similarly for a subset of languages, and if so, how do we cluster different languages? How do we incorporate techniques for languages that may not have much in common with other languages? Such and other questions can benefit from further investigation by future researchers.

BIBLIOGRAPHY

[1] Abras, C., Maloney-Krichmar, D., and Preece, J. (2004). User-centered design. In *W. Bainbridge (Ed.)*, Encyclopedia of Human-Computer Interaction, Sage Publications, Thousand Oaks.

[2] Acosta, G., Gonzalez, C., and Pulido, B. (2001). Basic tasks for knowledge-based supervision in process control. In *Engineering Applications of Artificial Intelligence*, 14 (4), pp. 441–455.

[3] Adda-Decker, M., and Lamel, L. (2005). Do speech recognizers prefer female speakers? In *Proc. InterSpeech,* Lisbon, Portugal*,* ISCA, pp. 2205–2208.

[4] Aist, G., Allen, J., Campana, E., Gallo, C., Stoness, S., Swift, M., and Tanenhaus, M. (2007). Incremental understanding in human-computer dialogue and experimental evidence for advantages over non-incremental methods. In *Proc. Workshop on Semantics and Pragmatics of Dialogue (DECALOG)*, Roverto, Italy, pp. 149-154.

[5] Altmann, G., and Kamide, Y. (1999). Incremental interpretation at verbs: restricting the domain of subsequent reference. In *Cognition, 73*(3), Elsevier, pp. 247-264.

[6] AngularJS – Superheroic JavaScript MVW Framework, https://angularjs.org/, last accessed June 24, 2014.

[7] Apple – iOS7 – Siri, http://www.apple.com/ios/siri/, last accessed May 22, 2014.

[8] Arora, P. (2006). Karaoke for social and cultural change. In *Journal of Information, Communication and Ethics in Society (ICES),* 4(3), Paper 1, Troubador Publishing Ltd, pp. 121-130.

[9] August, D., Carlo, M. S., Dressler, C., and Snow, C. (2005). The critical role of vocabulary development for English language learners. In *Learning Disabilities Research & Practice,* 20*,* pp. 50-57.

[10] Azim Premji Foundation. (2004). Impact of computer aided learning on learning achievements: a study in Karnataka and Andhra Pradesh.

[11] Babel – Intelligence Advanced Research Projects Activity, IARPA BAA-11-02, http://www.iarpa.gov/index.php/research-programs/babel, 2011, last accessed July 7, 2014.

[12] Bali, K., Sitaram, S., Cuendet, S., and Medhi, I. (2013). A hindi speech recognizer for an agricultural video search application. In *Proc. Symposium on Computing for Development (DEV)*, Bangalore, India, ACM, pp. 5.

[13] Banerjee, A., Cole, S., Duflo, E., and Lindon, L. (2007). Remedying education: evidence from two randomized experiments in India. In *Quarterly Journal of Economics*, 122(3), pp. 1235-64.

[14] Baumann, T., Atterer, M., and Schlangen, D. (2009). Assessing and improving the performance of speech recognition for incremental systems. In *Proc. HLT NAACL*, Boulder, USA, ACL, pp. 380-388.

[15] Bisani, M., and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. In *Speech Communication*, 50(5), Elsevier, pp. 434 – 451.

[16] Bellegarda, J. (2004). Statistical language model adaptation: Review and perspectives. *Speech Communication,* 42(1), Elsevier, pp. 93-108.

[17] Bourbakis, N. G., Mogzadeh, A., Mertoguno, S., and Koutsougeras, C.A. (2002). A knowledge-based expert system for automatic visual VLSI reverse-engineering: VLSI layout version. In *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 32(1), IEEE, pp. 428–437.

[18] Callison-Burch, C., and Dredze, M. (2010). Creating speech and language data with Amazon's mechanical turk. In *Proc. NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, Los Angeles, USA, ACL, pp. 1-12.

[19] Carnegie Speech, http://www.carnegiespeech.com/, last accessed May 22, 2014.

[20] Chen, C.P., Filaliy, K., and Bilmes, J.A. (2002). Frontend post-processing and backend model enhancement on the Aurora 2.0/3.0 databases. In *Proc. InterSpeech*, Denver, USA, ISCA.

[21] Chu, S., and Povey, D. (2010). Speaking rate adaptation using continuous frame rate normalization. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* Dallas, USA, IEEE, pp. 4306-4309.

[22] Clancy, P., Gerald H., and Arnold S. (1989). An expert system for legal consultation. In *Proc. 2nd Annual Conference on Innovative Applications of Artificial Intelligence*, Menlo Park, USA, AAAI, pp. 125-135.

[23] Clark, H.H., and Brennan, S.E. (1991). Grounding in communication. In *Perspectives on Socially Shared Cognition*, APA, pp. 127-149.

[24] Compton, P. (2013). Situated cognition and knowledge acquisition research. In *International Journal of Human-Computer Studies*, 71(2), ACM, pp. 184-190.

[25] Crossfilter, http://square.github.io/crossfilter/, last accessed June 24, 2014.

[26] d3.js – Data-Driven Documents, http://d3js.org/, last accessed June 24, 2014.

[27] Dahl, G., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. In *Transactions on Audio, Speech, and Language Processing*, 20(1), IEEE, pp. 30-42.

[28] Dani, M.N., Faruquie, T.A., Garg, R., Kothari, G., Mohania, M.K., Prasad, K.H., Subramaniam, L.V., and Swamy, V.N. (2010). Knowledge acquisition method for improving data quality in services engagements. In *Proc. International Conference on Services Computer*, Miami, USA, IEEE, pp. 346–353.

[29] Davel, M., and Martirosian, O. (2009). Pronunciation dictionary development in resource-scarce environments. In *Interspeech*, Brighton, UK, ISCA, pp. 2851-2854.

[30] dc.js – Dimensional Charting Javascript Library, http://dc-js.github.io/dc.js/, last accessed June 24, 2014.

[31] De Bot, K. (1996). The psycholinguistics of the output hypothesis. In *Language Learning,* 46(3)*,* pp. 529-555.

[32] Demšar, J., Zupan, B., Leban, G., and Curk, T. (2004). Orange: from experimental machine learning to interactive data mining. In *Knowledge Discovery in Databases (KDD),* Pisa, Italy, Springer, pp. 537-539.

[33] Dias, M.B., Mills-Tettey, G.A., and Mertz, J. (2005). The TechBridgeWorld initiative: broadening perspectives in computing technology education and research. In *Proc. International Symposium on Women and ICT: Creating Global Transformation*, Baltimore, USA, ACM, Article No. 17.

[34] Dirks, S., Kingston, J.K.C., and Haggith, M. (1995). Development of a KBS for personal financial planning guided by pragmatic KADS. In *Expert Systems with Applications*, 9(2), AIAI, pp. 91–101.

[35] Dunn, L.M., and Dunn, L.M. (1997). Peabody Picture Vocabulary Test (3rd edition). In *Circle Pines,* MN, USA*,* American Guidance Service.

[36] Eskenazi, M. (2009). An overview of spoken language technology for education. In *Speech Communication*, 51(10), Elsevier, pp. 832-844.

[37] Eskenazi, M., Mostow, J., and Graff, D. (1997). The CMU kids speech corpus. Corpus of children's read speech digitized and transcribed on two CD-ROMs, with assistance from Multicom Research and David Graff. In *Linguistic Data Consortium*, University of Pennsylvania.

[38] Fails, J.A. and Olsen, D. (2003). A design tool for camera-based interaction. In *Proc. Human Factors in Computing* (*CHI)*, Fort Lauderdale, USA, ACM, pp. 449-456.

[39] Fink, G., Schillo, C., Kummert, F., and Sagerer, G. (1998). Incremental speech recognition for multimodal interfaces. In *Proc. Industrial Electronics Society (IECON)*, IEEE, pp. 2012-2017.

[40] Fogarty, J., Tan, D., Kapoor, A., and Winder, S. (2008). CueFlik: Interactive Concept Learning in Image Search. In *Proc. Human Factors in Computing* (*CHI)*, Florence, Italy, ACM, pp. 29-38.

[41] Fosler-Lussier, E., and Morgan, N. (1999). Effects of speaking rate and word frequency on pronunciations in conversational speech. In *Speech Communications*, 29(2-4), Elsevier, pp. 137–158.

[42] Freitas, J., Calado, A., Braga, D., Silva, P., and Dias, M. (2010). Crowdsourcing platform for large-scale speech data collection. In *Proc. FALA,* Vigo, Spain, ISCA.

[43] Gales, M.J.F. (1998). Maximum likelihood linear transformations for HMM-based speech recognition. In *Computer Speech and Language,* 12(2), Elsevier, pp. 75-98.

[44] Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., and Zue, V. (1993). TIMIT acoustic-phonetic continuous speech corpus, http://ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1, last accessed June 14, 2013.

[45] Gee, J.P. (2003). What video games have to teach us about learning and literacy. First edition. *Palgrave Macmillan*.

[46] Giuliani, D., and Gerosa, M. (2003). Investigating recognition of children's speech. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* Hongkong, IEEE, pp. 137-140.

[47] Goldwater, S., Jurafsky, D., and Manning, C. (2010). Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates. In *Speech Communication*, 52(3), Elsevier, pp. 181-200.

[48] Google Now – Voice Search, http://www.google.com/insidesearch/features/voicesearch/index.html, last accessed February 15, 2013.

[49] Google Speech API, https://www.google.com/intl/en/chrome/demos/speech.html, last accessed July 30, 2014.

[50] Goronzy, S., and Ralf K. (1999). A combined MAP+ MLLR approach for speaker adaptation. In *Proc. Sony Research Forum*, Tokyo, Japan, Sony, volume 1, pp. 9-14.

[51] Graff, D., Walker, K., and Miller, D. (2004). Switchboard cellular part 2 audio. In *Linguistic Data Consortium*.

[52] Grover, D.L., King, M., and Kushler, C.A. (1998). Patent No. US5818437, reduced keyboard disambiguating computer. By *Tegic Communications Inc.*, Seattle.

[53] Gunawardana, A., Paek, T., and Meek, C. (2010). Usability guided key-target resizing for soft keyboards. In *Proc. Intelligent User Interfaces (IUI)*, Madeira, Portugal, ACM, pp. 111-118.

[54] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H. (2009). The WEKA data mining software: an update. In *SIGKDD Explorations*, 11(1), ACM, pp. 10-18.

[55] Haberman, S.J. (1979). Analysis of qualitative data: new developments. Volume 2. By *Academic Press*, New York, USA.

[56] Hagen, A., Pellom, B., and Cole, R. (2007). Highly accurate children's speech recognition for interactive reading tutors using subword units. In *Speech Communication*, 49(12), Elsevier, pp. 861-873.

[57] Hart, S.G. (2006). Nasa-task load index (Nasa-TLX): 20 years later. In *Proc. Human Factors and Ergonomics Society Annual Meeting (HFES)*, IEA, pp. 904-908.

[58] Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S.R. (2007). Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proc. Human Factors in Computing Systems (CHI)*, San Jose, USA, ACM, pp. 145-154.

[59] Hatzilygeroudis, I., and Prentzas, J. (2004). Using a hybrid rule-based approach in developing an intelligent tutoring system with knowledge acquisition and update capabilities. In *Expert Systems with Applications: An International Journal*, 26(4), ACM, pp. 477–492.

[60] Hirschberg, J., Litman, D., and Swerts, M. (2004). Prosodic and other cues to speech recognition failures. In *Speech Communications*, 43(1-2), Elsevier, pp. 155–175.

[61] Holmes, J. (1992). An introduction to sociolinguistics. By *Longman Group,* London, UK.

[62] HTK Speech Recognition Toolkit, http://htk.eng.cam.ac.uk/, last accessed December 16, 2011.

[63] Huggins-Daines, D., Kumar, M., Chan, A., Black, A., Ravishankar, M., and Rudnicky, A.I. (2006). PocketSphinx: a free, real-time continuous speech recognition system for handheld devices. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, IEEE, pp. 185–188.

[64] Hura, S. (2008). Voice user interfaces. In *Kortum, P. (Ed.)*, HCI Beyond the GUI: Design for Haptic, Speech, Olfactory, and other Nontraditional Interfaces, by Morgan Kaufman Publishers.

[65] Junqua J.C. (1993). The Lombard reflex and its role on human listeners and automatic speech recognizers. In *Journal of the Acoustical Society of America (JASA),* 93(1), pp. 510–524.

[66] Kam, M., Agarwal, A., Kumar, A., Lal, S., Mathur, A., Tewari, A., and Canny, J. (2008). Designing e-learning games for rural children in India: A format for balancing learning with fun. In *Proc. Designing Interactive Systems (DIS)*, Cape Town, South Africa, ACM, pp. 58-67.

[67] Kam, M., Kumar, A., Jain, S., Mathur, A., and Canny, J. (2009). Improving literacy in rural India: cellphone games in an after-school program. In *Proc. Information and Communication Technology and Development (ICTD)*, Doha, Qatar, IEEE/ACM, pp. 139-149.

[68] Kam, M., Mathur, A., Kumar, A., and Canny, J. (2009). Designing digital games for rural children: A study of traditional village games in India. In *Proc. Human Factors in Computing Systems (CHI),* Boston, USA, ACM, pp. 31-40.

[69] Kam, M., Ramachandran, D., Devanathan, V., Tewari, A., and Canny, J. (2007). Localized iterative design for language learning in underdeveloped regions: The PACE framework. In *Proc. Human Factors in Computing Systems (CHI*), San Jose, USA, ACM, pp. 1097-1106.

[70] Karat, C.M., Halverson, C., Karat, J., and Horn, D. (1999). Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proc. Human Factors in Computing Systems (CHI)*, Pittsburgh, USA, ACM, pp. 568-575.

[71] Kaldi, http://kaldi.sourceforge.net/, last accessed May 22, 2014.

[72] Kapoor, A., Lee, B., Tan, D., and Horvitz, E. (2010). Interactive optimization for steering machine classification. In *Proc. Human Factors in Computing Systems* (*CHI),* Atlanta, USA, ACM, pp. 1343-1352.

[73] Klemmer, S., Sinha, A., Chen, J., Landay, J., Aboobaker, N., and Wang, A. (2001). SUEDE: A wizard of oz prototyping tool for speech user interfaces. In *Proc. Symposium on User Interface Software and Technology (UIST)*, San Diego, USA, ACM, pp. 1-10.

[74] Kneser, R., and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Detroit, USA, pp. 181–184.

[75] Kotelly, B. (2003). The Art and Business of Speech Recognition: Creating the Noble Voice. In *Addison-Wesley Longman Publishing Co.*, Boston, USA.

[76] Kumar, A., Metze, F., Wang, W., and Kam, M. (2013). Formalizing expert knowledge for developing accurate speech recognizers. In *Proc. InterSpeech*, Lyon, France, ISCA, pp. 1121-1125.

[77] Kumar, A., Paek, T., and Lee, B. (2012). Voice Typing: A new speech interaction model for dictation on touchscreen devices. In *Proc. Human Factors in Computing Systems (CHI)*, Austin, Texas, ACM, pp. 2277-2286.

[78] Kumar, A., Reddy, P., Tewari, A., Agrawal, R., and Kam, M. (2012). Improving literacy in developing countries using speech recognition-supported games on mobile devices. In *Proc. Human Factors in Computing Systems (CHI)*, Austin, Texas, ACM, pp. 1149-1158.

[79] Kumar, A., Tewari, A., Horrigan, S., Kam, M., Metze, F., and Canny, J. (2011). Rethinking speech recognition on mobile devices. In *Proc. 2nd International Workshop on*

Intelligent User Interfaces for Developing Regions (IUI4DR) with IUI, Menlo Park, USA, ACM.

[80] Kumar, A., Tewari, A., Shroff, G., Chittamuru, D., Kam, M., and Canny, J. (2010). An exploratory study of unsupervised mobile learning in rural India. In *Proc. Human Factors in Computing Systems (CHI)*, Atlanta, Georgia, ACM, pp. 743-752.

[81] Kurtenbach, G., and Buxton, W. (1994). User learning and performance with marking menus. In *Proc. Human Factors in Computing Systems (CHI)*, Boston, USA, ACM, pp. 258-264.

[82] Lai, J., Clare-Marie, K., and Yankelovich, N. (2008). Conversational Speech Interfaces and Technologies. In Andrew Sears and Julie Jacko (Eds.), *The Human-Computer Interaction Handbook*, 2nd Edition, Taylor and Francis,.

[83] Laput, G., Dontcheva, M., Wilensky, G., Chang, W., Agarwala, A., Linder, J., & Adar, E. (2013). PixelTone: a multimodal interface for image editing. In *Proc. Human Factors in Computing Systems (CHI)*, Paris, France, ACM, pp. 2185-2194.

[84] Lee, C.H., and Gauvain, J.L. (1992). MAP estimation of continuous density HMM: theory and applications. In *Proc. DARPA Speech & Natural Language Workshop with HLT*, Pacific Grove, USA, ACM, pp. 185-190.

[85] Levenshtein V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady,* 10(8), pp. 707-710.

[86] Liao, S. H. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. In *Expert Systems with Applications*, 28(1), Elsevier, pp. 93-103.

[87] LMSharp, http://lmsharp.codeplex.com/, last accessed June 16, 2014.

[88] Lockwood, P., and Boudy, J. (1992). Experiments with a non-linear spectral subtractor, HMM's and the projection, for robust speech recognition in cars. In *Speech Communications,* 11(2-3), Elsevier, pp. 215-228.

[89] Lopes, J., Eskenazi, M., and Trancoso, I. (2013) Automated two-way entrainment to improve spoken dialog system performance. In *International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP)*, Vancouver, Canada, IEEE, pp. 8372-8376.

[90] Luce, P., and Pisoni, D. (1998). Recognizing spoken words: the neighborhood activation model. In *Ear and Hearing*, 19(1), American Auditory Society, pp. 1–36.

[91] MacKenzie, I.S., and Soukoreff, R.W. (2002). Text entry for mobile computing: Models and methods, theory and practice. In *Journal on Human-Computer Interaction* 17(2-3), ACM, pp. 147-198.

[92] Martirosian, O.M. and Davel, M. (2007). Error analysis of a public domain pronunciation dictionary. In *Proc. 18th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, Pietermaritzburg, South Africa, ACM, pp. 13–18.

[93] Martin, T.B., and Welch, J.R. (1980). Practical speech recognizers and some performance effectiveness parameters. In *Trends in Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ, USA.

[94] Maynes-Aminzade, D., Winograd, T., and Igarashi, T. (2007). Eyepatch: prototyping camera-based interaction through examples. In *Proc. Symposium on User Interface Software and Technology (UIST)*, Newport, USA, ACM, pp. 33-42.

[95] McCandless, M. (1992). Word rejection for a literacy tutor. *B.S. Thesis.* Department of Electrical Engineering and Computer Science, MIT.

[96] Metze, F., Fosler-Lussier, E., and Bates, R. (2013). The Speech Recognition Virtual Kitchen. In *Proc. InterSpeech*, Lyon, France, ISCA.

[97] Miao, Y. (2014). Kaldi+PDNN: Building DNN-based ASR Systems with Kaldi and PDNN. arXiv:1401.6984.

[98] Microsoft Cortana – Meet Cortana for Windows Phone, http://www.windowsphone.com/en-us/how-to/wp8/cortana/meet-cortana, last accessed July 30, 2014.

[99] Microsoft Speech API, http://msdn.microsoft.com/en-us/library/ms720151%28v=vs.85%29.aspx, last accessed July 30, 2014.

[100] Miller, G.A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. In *Psychological Review*, 63(2), APA, pp. 81-97.

[101] Mills-Tettey, A., Mostow, J., Dias, M.B., Sweet, T.M., Belousov, S.M., Dias, M.F., and Gong, H. (2009). Improving child literacy in Africa: experiments with an automated reading tutor. In *Proc. Information and Communication Technologies on Development (ICTD)*, Doha, Qatar, IEEE/ACM, pp. 129-138.

[102] Munteanu, C., Penn, G., Baecker, R., and Zhang, Y. (2006). Automatic speech recognition for webcasts: how good is good enough and what to do when it isn't? In *Proc. International Conference on Multimodal Interaction (ICMI)*, Banff, Canada, ACM, pp. 39-42.

[103] Munteanu, C., Jones, M., Oviatt, S., Brewster, S., Penn, G., Whittaker, S., Rajput, N., and Nanavati, A. (2013). We need to talk: HCI and the delicate topic of spoken language interaction. In *Proc. Human Factors in Computing Systems (CHI)*, Paris, France, ACM, pp. 2459-2464.

[104] Munteanu, C., Jones, M., Whittaker, S., Oviatt, S., Aylett, M., Penn, G., Brewster, S., and d'Alessandro. (2014). Designing speech and language interactions. In *Proc. Workshop on Designing Speech and Language Interactions at Human Factors in Computing Systems (CHI)*, Toronto, Canada, ACM, pp. 75-78.

[105] NIST Speech Quality Assurance Package, http://www.icsi.berkeley.edu/Speech/papers/gelbart-ms/pointers/#NIST, last accessed July 30, 2014.

[106] Nuance – Dragon Dictation: iPhone – Dragon Dictation for iPad™, iPhone™ and iPod touch™ is an easy-to-use voice recognition application. http://www.nuance.com/for-business/by-product/dragon-dictation-iphone/index.htm, last accessed May 27, 2014.

[107] O'Connor, M., Knublauch, H., Tu, S.W., and Musen, M.A. (2005). Writing Rules for the Semantic Web using SWRL and Jess. In *Proc. Protégé With Rules Workshop*, Madrid, Spain.

[108] Olive, J., Caitlin, C., John, M (eds.). (2010). Handbook of Natural Language Processing and Machine Translation. In *DARPA Global Autonomous Language Exploitation (GALE)*, Springer.

[109] Oviatt, S. (2003). Advances in Robust Multimodal Interface Design. In *IEEE Computing Graph Application*, IEEE, pp. 23-25.

[110] Paul, D., and Baker, J. (1992). The design for the wall street journal-based CSR corpus. In *Proc. Human Language Technologies (HLT)*, Pacific Grove, USA, ACL, pp. 357-362.

[111] Paek, T., Ju, Y. C., and Meek, C. (2007). People watcher: a game for eliciting human-transcribed data for automated directory assistance. In *Proc. InterSpeech,* Antwerp, Belgium, ISCA, pp. 1322-1325.

[112] Parent, G., and Eskenazi, M. (2011). Speaking to the crowd: looking at past achievements in using crowdsourcing for speech and predicting future challenges. In *Proc. InterSpeech,* Florence, Italy, ISCA, pp. 3037-3040.

[113] Patel, K., Bancroft, N., Drucker, S.M., Fogarty, J., Ko, A., and Landay, J.A. (2010). Gestalt: integrated support for implementation and analysis in machine learning processes. In *Proc. Symposium on User Interface Software and Technology (UIST)*, New York, USA, ACM, pp. 37-46,.

[114] Pawar, U., Pal, J., Gupta, R., and Toyama, K. (2007). Multiple mice for retention tasks in disadvantaged schools. In *Proc. Human Factors in Computing Systems (CHI)*, San Jose, USA, ACM, pp. 1581-1590.

[115] Payne, S. (2009). Mental models in human-computer interaction. In *The Human-Computer Interaction Handbook*, Lawrence Erlbaum, Mahwah, New Jersey, USA.

[116] Penn, G., and Zhu, X. (2008). A critical reassessment of evaluation baselines for speech summarization. In *Proc. Human Language Technologies (HLT),* Columbus, USA, ACL, pp. 470-478.

[117] Perfetti, C. (2007). Reading ability: lexical quality to comprehension. In *Scientific Studies of Reading,* 11(4)*,* pp. 357-383.

[118] Perfetti, C. A., and Hart. L. (2001). The lexical quality hypothesis. In *L. Verhoeven, C. Elbro & P. Reitsma (Eds.), Precursors of Functional Literacy,* 11, Amsterdam: John Benjamins, pp. 67–86.

[119] Plauchè, M., Nallasamy, U., Wooters, C., Ramachandran, D., and Pal, J. (2006). Speech recognition for illiterate access to information and technology. In *Proc. International Conference on Information and Communication Technologies and Development (ICTD)*, Berkeley, USA, IEEE, pp. 83-92.

[120] Praat: doing phonetics by computer, http://www.fon.hum.uva.nl/praat/, last accessed May 22, 2014.

[121] Preece, J., Rogers, Y., and Sharp, H. (2002). In *Interaction design: Beyond human-computer interaction*, John Wiley and Sons, New York, USA.

[122] Qin, L. (2003). Learning Out-of-Vocabulary Words in Automatic Speech Recognition. *PhD Thesis CMU-LTI-13-014.* Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

[123] Rabiner, L., and Juang, B.H. (1993). Fundamentals of speech recognition. By *Prentice Hall*, Upper Saddle River, New Jersey, USA.

[124] Rabiner L.R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proc. IEEE,* 77(2), IEEE, pp. 257-286.

[125] Raza, A.A., Haq, F., Tariq, Z., Pervaiz, M., Razaq, S., Saif, U., and Rosenfeld, R. (2013). Job opportunities through entertainment: virally spread speech-Based services for low-literate users. In *Proc. Human Factors in Computing Systems (CHI)*, Paris, France, ACM, pp. 350-359.

[126] Riccardi, G., and Hakkani-Tur, D. (2005). Active learning: theory and applications to automatic speech recognition. In *Speech and Audio Processing, IEEE Transactions on Audio, Speech, and Language Processing*, 13(4), IEEE, pp. 504-511.

[127] Richards, D. (2009). Two decades of ripple down rules research. In *Knowledge Engineering Review*, 24(2), ACM, pp. 159-184.

[128] Rodionov, S.N., and Martin, J.H. (1999). An expert system-based approach to prediction of year-to-year climatic variations in the north Atlantic region. In *International Journal of Climatology*, 19(9), Wiley Publishers, pp. 951–974.

[129] Ronanki, S., Bo, L., and Salsman, J. (2012). Automatic pronunciation evaluation and mispronunciation detection using CMU sphinx. In *Proc. Speech and Language Processing Workshop for Tools in Education (SLP-TED) with COLING*, Mumbai, India, ACL, pp.61-68.

[130] Rosen, L.D. (1997). A review of speech recognition software. In *The National Psychologist*, 6(5), pp. 28-29.

[131] Rosetta Stone, Ltd. Rosetta Stone Language Learning, http://www.rosettastone.com/schools, last accessed May 27, 2014.

[132] Scikit-learn: Machine Learning in Python, http://scikit-learn.org/stable/, last accessed July 1, 2014.

[133] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B. (2000). Knowledge engineering and management: the commonKADS methodology. In *MIT Press*, Cambridge, USA.

[134] Schultz, T., Black, A., Badaskar, S., Hornyak, M., and Kominek, J. (2007). SPICE: web-based tools for rapid language adaptation in speech processing systems. In *Proc. Interspeech*, Antwerp, Belgium, ISCA.

[135] Selfridge, E., Arizmendi, I., Heeman, P., and Williams, J. (2011). Stability and accuracy in incremental speech recognition. In *Proc. Special Interest Group on Discourse and Dialogue (SIGDIAL)*, Portland, USA, ACL, pp. 110-119.

[136] Sen, M.D.L., Minambres, J.J., Garrido, A.J., Almansa, A., and Soto, J.C. (2004). Basic theoretical results for expert systems, application to the supervision of adaptation transients in planar robots. In *Artificial intelligence*, 152(2), Elsevier, pp. 173–211.

[137] Sherwani, J., Palijo, S., Mirza, S., Ahmed, T., Ali, N., and Rosenfeld, R. (2009). Speech vs. touch-tone: telephony interfaces for information access by low literate users. In *Proc. Information & Communications Technologies and Development (ICTD)*, Doha, Qatar, ACM, pp. 447-457.

[138] Shinoda, K. (2011). Speaker adaptation techniques for automatic speech recognition. In *Proc. Asia-Pacific Signal and Information Processing Association (APSIPA) Annual Summit and Conference,* Xi'an, China.

[139] Shinozaki, T., and Furui, S. (2001). Error analysis using decision trees in spontaneous presentation speech recognition. In *Proc. Automatic Speech Recognition and Understanding (ASRU)*, Madonna di Campiglio, Italy, IEEE, pp. 198-201.

[140] Shpilberg, D., Graham, L., and Schatz, H. (2012). ExperTAX^{sm}: An expert system for corporate tax planning. In *Expert Systems: The Journal of Knowledge Engineering*, 3(2), ACM, pp. 136–151.

[141] Shobaki, K., Hosom, J.P., and Cole, R. (2007). CSLU: Kids' speech version 1.1. In *Linguistic Data Consortium*, http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2007S18, last accessed August 3, 2014.

[142] Shobaki, K., Hosom, J.P., and Cole, R. (2000). The OGI kids' speech corpus and recognizers. In *Proc. International Conference on Spoken Language Processing (ICSLP),* Beijing, China, ISCA.

[143] Shortliffe, E.H. (1976). Computer-based medical consultations: MYCIN. By *Elsevier*, New York, USA.

[144] Siegler, M., and Stern, R. (1995). On the effects of speech rate in large vocabulary speech recognition systems. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Detroit, USA, IEEE, pp. 612-615.

[145] Skantze, G., and Schlangen, D. (2009). Incremental dialogue processing in a micro-domain. In *Proc. 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Athens, Greece, ACL, pp. 745-753.

[146] Sphinx, http://sourceforge.net/p/cmusphinx/discussion/, last accessed May 22, 2014.

[147] Steele, J., and To, N. (2010). The android developer's cookbook: building applications with the android SDK. 1st Edition. By *Addison-Wesley Professional*.

[148] Steidl, S., Stemmer, G., Hacker, C., Nöth, E., and Niemann, H. (2003). Improving children's speech recognition by HMM interpolation with an adults' speech recognizer. In *Pattern Recognition,* Springer Berlin Heidelberg, pp. 600-607.

[149] Steinmann, H., and Chorafas, D.N. (1990). Expert systems in banking: a guide for senior managers. By *New York University Press*, New York, USA, pp. 222–225.

[150] Stifelman, L., Elman, A., and Sullivan, A. (2013). Designing natural speech interactions for the living room. In *Proc. Extended Abstracts on Human Factors in Computing Systems (CHI),* Paris, France, ACM, pp. 1215-1220.

[151] Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. In *Proc. International Conference on Spoken Language Processing (ICSLP)*, Denver, Colorado, ISCA, pp. 901-904.

[152] Strauss, A.C., and Corbin, J. (2008). Basics of qualitative research: techniques and procedures for developing grounded theory. 3rd Edition. By *Sage Publications*.

[153] Suhm, B., Myers, B., and Waibel, A. (2001). Multi-modal error correction for speech user interfaces. In *Transaction of Human Factors in Computing Systems (TOCHI),* 8(1), ACM, pp. 60-98.

[154] Swain, M., and Lapkin, S. (1995). Problems in output and the cognitive processes they generate: a step towards second language learning. In *Applied Linguistics*, 16(3), Oxford University Press, pp. 371-391.

[155] SWYPE | Type Fast, Swype Faster, http://www.swype.com, last accessed May 22, 2014.

[156] Talbot, J., Lee, B., Kapoor, A., and Tan, D. (2009). EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In *Proc. Human Factors in Computing Systems (CHI)*, Boston, USA, ACM, pp. 1283-1292.

[157] Tanenhaus, M.K., Spivey-Knowlton, M.J., Eberhard, K.M., & Sedivy, J.C. (1995). Integration of visual and linguistic information in spoken language comprehension. In *Science,* 268(5217), AAAS, pp. 1632-1634.

[158] Tewari, A. (2013). Speech-enabled Systems for Language Learning. *PhD Thesis UCB/EECS-2013-67*. EECS Department, University of California, Berkeley.

[159] Tewari, A., Goyal, N., Chan, M., Yau, T., Canny, J., and Schroeder, U. (2010). SPRING: speech and pronunciation improvement through games, for hispanic children. In *Proc. Information and Communication Technologies and Development (ICTD)*, London, UK, IEEE/ACM, Article No. 47.

[160] The Importance of Adaptation in Speech Recognition, http://www.speechtechmag.com/Articles/Editorial/Sounding-Board/The-Importance-of-Adaptation-in-Automatic-Speech-Recognition-68360.aspx, published July 16, 2010, last accessed May 22, 2014.

[161] Thomason, J., and Litman, D. (2013). Differences in User Responses to a Wizard-of-Oz versus Automated System. In *Proc. NAACL HLT*, Atlanta, USA, ACL, pp. 796-801.

[162] Traxler, M.J., Bybee, M.D., and Pickering, M.J. (1997). Influence of connectives on language comprehension: eye-tracking evidence for incremental interpretation. In *The Quarterly Journal of Experimental Psychology: Section A,* 50(3), pp. 481-497.

[163] Van Doremalen, J., Helmer S., and Catia C. (2010). Speech technology in CALL: the essential role of adaptation. In *Proc. Interdisciplinary Approaches to Adaptive Learning: A Look at the Neighbours: First International Conference on Interdisciplinary Research on Technology, Education and Communication (ITEC)*, Kortrijk, Belgium, Springer, vol. 126, pp. 56-69.

[164] Van Ess-Dykema, C., and Ries, K. (1998). Linguistically engineered tools for speech recognition error analysis. In *Proc. International Conference on Spoken Language Processing (ICSLP)*, Sydney, Australia, ISCA, pp. 2091-2094.

[165] Vergin, R., Farhat, A., and O'Shaughnessy, D. (1996). Robust gender-dependent acoustic-phonetic modeling in continuous speech recognition based on a new automatic male/female classification. In *Proc. International Conference on Spoken Language Processing (ICSLP)*, Philadelphia, USA, ISCA, pp. 1081-1084.

[166] Voice to Text Application Powered by Intelligent Voice Recognition | Vlingo, http://www.vlingo.com, last accessed August 3, 2014.

[167] Watanabe, S., and Le Roux, J. (2014). Black box optimization for automatic speech recognition. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, IEEE, pp. 3256-3260.

[168] Windows 7 Speech Recognition | Voice Recognition, http://www.microsoft.com/windowsphone/en-us/howto/wp7/basics/use-speech-on-my-phone.aspx, last accessed May 27, 2014.

[169] Witten, I.H. and Frank, E. (2005). Data mining: practical machine learning tools and techniques. 2nd Edition. By *Morgan Kaufmann Publishers In,* Burlington, USA.

[170] Woodland, P. C. (2001). Speaker adaptation for continuous density HMMs: A review. In *Proc. Workshop on Adaptation Methods for Speech Recognition,* Sophia-Antipolis, France, ISCA, pp. 11–19.

[171] Yee, C., and Rosenfeld, R. (2012). Discriminative pronunciation learning for speech recognition for resource scarce languages. In *Proc. 2nd Symposium on Computing for Development (DEV)*, London, UK, ACM, Article No. 12.

[172] Zhai, S., and Kristensson, P.O. (2003). Shorthand writing on stylus keyboard. In *Proc. Human Factors in Computing Systems (CHI)*, Fort Lauderdale, USA, ACM, pp. 97-104.