Enabling the Rapid Development and Adoption of Speech-User Interfaces

Anuj Kumar and Florian Metze, *Carnegie Mellon University*
Matthew Kam, *American Institutes for Research*

Speech-user interfaces offer truly hands-free, eyes-free interaction, have unmatched throughput rates, and are the only plausible interaction modality for illiterate users across the world, but they are not yet developed in abundance to support every type of user, language, or acoustic scenario. Two approaches present exciting opportunities for future research.

Speech-based user interfaces have become increasingly popular, especially since the introduction of Apple's Siri in 2010. Google and Samsung quickly followed suit with their own speech-driven services called Google Now and S Voice, respectively. The boom in speech-recognition applications is not surprising, given that they offer several potential advantages. First, speech-based interaction offers truly hands-free, eyes-free interaction, a dream that has evaded us for many years. Second, speech is faster than typing on a keyboard, and without the need for an onscreen keyboard, there is much greater flexibility in terms of screen real estate. Finally, speech-driven applications present important opportunities for the 800 million or so illiterate users in developing regions, giving them a feasible way to access computing.

However, beyond a few success stories, we have not yet seen large numbers of functional and sufficiently accurate speech-recognition services. There are several reasons for this, although two key challenges in particular stand out.

First, to deploy a usable speech recognizer, developers must build something that can be optimized for every new user group, language, or acoustic situation; and from the developer's perspective, because he or she is not typically an expert in speech recognition or acoustical engineering, this task is daunting.[1] Consequently, many application developers conduct Wizard of Oz studies, in which they simulate automatic speech recognition (ASR) instead of deploying a functional ASR system for the purpose of early experimentation or data collection. Unfortunately, simulated ASR cannot uncover real recognition errors,[2] which are the leading source of usability issues once the product is released.

Second, reaping real benefits of speech-recognition applications requires their successful deployment on mobile phones, or similar devices that users can interact with while on the go; however, most current mobile-based speech services require access to a reliable network connection and a data plan. For instance, Siri only works if you have high-speed network connectivity, which is usually unavailable in many rural regions of developing nations or in certain areas of developed ones, such as an underground subway station or inside a moving bus. One exception is Google, which has recently started to ship its general-purpose recognition models for local use on devices. However, these models aren't fine-tuned, even over time, to specific user or usage requirements for improved accuracy.

 Published by the IEEE Computer Society 0018-9162/14/$31.00 © 2014 IEEE

Nevertheless, ASR research has made great strides in recent years, reaching a point where, given enough speech expertise and in-domain data, a sufficiently accurate speech recognizer can be developed for any scenario; this includes cases where non-native accents, background noises, children's voices, or other similar challenges may be present. However, if a speech recognition system does not work as intended, it is generally impossible for a non-expert to pinpoint the exact reasons for failure. For example, recognition failure can result from a user speaking too slowly (speaking rate is a frequent cause of mismatch) or too clearly (if the person hyper-articulates), or if background noises are unexpected, or for some other reason, and these are error patterns that experts can usually analyze quickly. Adaptation, or several optimizations of an existing recognizer, can generally mitigate these errors, and will often result in a functional system,[3] but such adaptation requires that developers have substantial expertise and experience in speech recognition development. Application developers typically do not have such expertise or experience and find it difficult to identify people who do, which makes their task even more challenging.

Looking forward to where this technology is headed, we describe the design and development of both a speech toolkit that embeds expert knowledge into speech applications, as well as a new model for mobile device ASR systems that eliminates the requirement of a reliable network connection. Put together, these two innovations have the potential to solve the ASR's fundamental problems, thereby enabling rapid development and adoption of speech services in newer domains and languages, and for users whom it was not previously possible to assist.

A TOOLKIT FOR NON-EXPERTS

Several resources are currently available for non-expert speech-application developers. SUEDE, for example, lets any designer rapidly mock up a prompt-and-response speech interface for testing in a Wizard of Oz study.[4] It does not, however, support the development of a working recognizer. SPICE, on the other hand, supports rapid development of a baseline recognizer for new languages by allowing any researcher to input a set of audio files, corresponding phoneme set, and dictionary to generate the baseline acoustic model.[5] However, SPICE does not automatically perform acoustic- or language-specific optimizations, which are key to achieving reasonable accuracy.

Open source and commercial speech toolkits such as Sphinx (http://sourceforge.net/p/cmusphinx/discussion), Janus,[6] or Kaldi (http://kaldi.sourceforge.net) support both the development of a baseline recognizer and an adapted version. However, they do not provide any automatic guidance on what adaptations to perform, leaving it to the developer's expertise to understand the application's context and apply the appropriate improvement techniques. Consequently, non-expert researchers find these toolkits extremely difficult to use. In 2012 alone, a discussion forum for Sphinx had more than 6,000 posts with over 1,000 unique topics from non-experts asking for help on various issues related to speech recognition.

Industry leaders such as Google and Microsoft also offer various free APIs to facilitate integration of speech recognition into their applications. For instance, applications can send an audio file to a preconfigured server and receive a decoded transcript in real time. Although this solution works well for typical speech applications—those developed for native speakers for use in clean, quiet environments—it is less robust in contexts where the acoustics or language patterns slightly differ from those they were developed for. Moreover, developers cannot access the background acoustic and language models installed on the server to perform any adaptations that might be needed for the recognizer to work in their own application's scenario.

If a speech recognition system does not work as intended, it is generally impossible for a non-expert to pinpoint the exact reasons for failure.

We turned to speech experts to learn how they build accurate and usable speech interfaces. These experts are well trained with years of experiential knowledge that guides them intuitively in building recognizers for new languages, acoustic situations, or users. This knowledge is undoubtedly hard to transfer to non-experts directly, but by observing these experts in action, we can study and formalize their tacit knowledge, and build a toolkit for non-experts. This formalized knowledge can then be used to help novices in automatic analysis and to recommend appropriate optimization techniques.

To do this, we interviewed five experts at Carnegie Mellon University. First, we asked them to describe a general adaptation process, the common challenges they face, the people they consult, and the tools they use. In a second phase, we observed them in action on an actual 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. We then asked the expert to explain the steps (similar to a retrospective) that he or she would take to build the best recognizer for this dataset. The transcripts of these interviews became the basis for a line-by-line open-coding process to identify relevant concepts and themes that enhanced our understanding of the optimization process and the associated intuition.

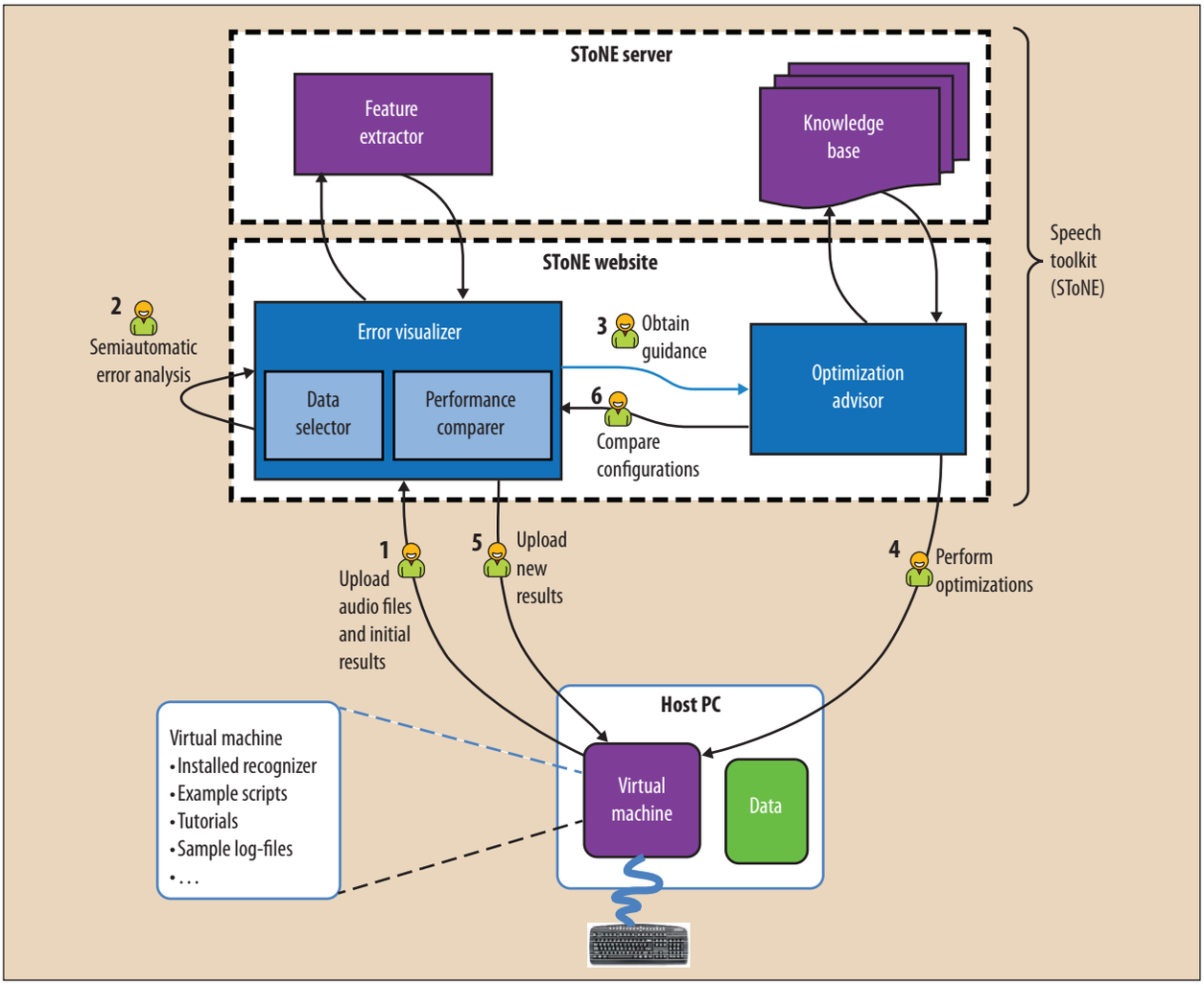

Figure 1. Process-flow diagram of the developer's interaction with the SToNE toolkit's modules. The numbers indicate the developer's sequence of steps, and the arrows indicate the direction of information flow. The developer uses a preconfigured virtual machine to conduct all experiments and interact with the website to avoid any hassles of recognizer installation.

Based on these interviews, we simplified our goals as follows:

- goal 1—come to a fine-grained understanding of why the speech recognizer is not working;
- goal 2—provide guidance to non-experts on steps they can take to make it work (including optimizations of the underlying recognizer and the user interaction style); and
- goal 3—enable performance visualization of competing setups to better understand the tradeoffs involved in each setup.

Accordingly, our speech toolkit for non-experts (or, as we call it, SToNE) has four modules: a feature extractor, an error visualizer, a knowledge base, and an optimization advisor. Figure 1 details how they are all tied together.

Feature extractor

The feature extractor's aim is to support goal 1—that is, help with analysis on why current recognizers might be failing or are unusable. To do so, this module extracts several lexical and prosodic features that correlate well with popular reasons for error such as pronunciation score, speaking rate, noise, and so on, and, by using regression analysis, it identifies the most significant features that impact recognition accuracy.

Error visualizer

The error visualizer supports goals 1 and 3 by helping the developer perform semiautomatic error analysis to understand recognition-error patterns. For a more detailed understanding of why the recognition might be failing, the error visualizer assists the user in doing three tasks: understanding the data distribution across variables, creating meaningful subsets and only analyzing that part of the data

(for example, all females under age 10, because of poor recognition accuracy), and comparing two or more subsets to understand differences between them. The latter would be particularly useful when comparing a subset giving high accuracy and a subset giving low accuracy to understand where exactly the two differ.

Table 1. Word-error percentage rates for the first test dataset.

Optimization technique	Experts					KB
	E1	E2	E3	E4	E5	
Baseline: use existing acoustic models (AMs)	94.7	94.7	94.7			94.7
Baseline: train new AMs				25.4	25.4	25.4
Maximum likelihood linear regression (MLLR) adaptation	77.3	77.3	77.3			
Maximum a posteriori (MAP) adaptation	78.3	78.3				
Vocal tract length normalization (VTLN)		76.9	76.9		24.9	
Adding pronunciation variants to dictionary			71.7	22.2		22.2
Frame rate adaptation						20.8
Final	**78.3**	**76.9**	**71.7**	**22.2**	**24.9**	**20.8**

The techniques in the first column were recommended by the experts (E) or the knowledge base (KB).

Knowledge base

The knowledge base (KB) supports goal 2 and consists of a set of rules extracted from an analysis of the interviews about the specific instructions or intuitions the experts had while analyzing a particular development situation. Upon formalization as "if, then, else" rules, the same experts vetted the KB for consistency and accuracy of formulation.

Table 2. Word-error rate percentages for second test dataset.

Optimization technique	E6	E7	KB
Add half-words to dictionary	✓	✓	✓
Correct spelling errors		✓	✓
Baseline: train new AM		56.1	56.1
Baseline: train new AM + model bootstrapping	42.2		
Cepstral variance normalization (CVN)	40.9		54.3
Frame-rate adaptation		54.8	52.2
Final	**40.9**	**54.8**	**52.2**

Optimization advisor

The optimization advisor also supports goal 2, acting as a front end to the KB for the developer. Given a specific acoustic situation, a new user group, or a new language, it queries the KB for the appropriate steps needed to build an accurate recognizer and communicates the answer to the developer. Once a step-by-step strategy has been recommended, the developer can then refer to tutorials in the virtual machine to obtain guidance on actually performing the optimizations and building the required speech recognizer or, in some cases, the interface itself.

Initial results

The crux of our work is in understanding whether expert knowledge can be properly formalized, and if so, the extent to which it can be used for the benefit of nonexperts. Our work so far has focused on answering this question and evaluating the KB's quality in terms of its ability to predict correct techniques for both seen and unseen circumstances.[7]

First, to see which rules get triggered and how well they actually model expert intuition, we evaluated a KB with the same dataset used in the expert interviews. Table 1 lists the results in terms of word-error rate improvements. On the whole, the KB's recommendations either mirrored or performed better than those of all five experts. The major point of difference arose in a step in which the experts themselves had conflicting opinions (step 1). To deal with such cases, we incorporated the conflicting alternatives from all experts with a priority ranking for each option. Initially, this priority ranking was the number of experts who had recommended a particular option, but, over time, other experts could provide additional feedback based on new test runs to either update these rankings or add more recent techniques to the KB.

Second, to see how well our KB performed for previously unseen datasets or users, we tested it on a second new dataset that contained speech from poor readers of English recorded in noisy environments. This dataset differed from the first by focusing on recognizing full sentences, not isolated words. For this evaluation, we hired two additional experts for their recommendation on techniques, as we also wanted to see how the techniques recommended by our KB compared with those of other experts who did not contribute to the KB's development. Table 2 shows the results.

On the second task, our KB outperformed one expert (E6) and underperformed another (E7); E7 recommended a technique that was not initially covered by the KB but was significantly better than the KB's alternatives. As the boundaries of speech recognition expand, such situations are likely to arise in the future as new techniques become available or old ones become obsolete. Fortunately, with

its rule-based design, the KB is easy to update with newer techniques.

Although the KB performed at least as well as any expert in the first task—and it can be expanded to incorporate the techniques from E7 for the second task—our work is limited to defining "if, then, else" rules based on datasets as presented to the seven experts. The KB would benefit from a more comprehensive approach than the current approach. Therefore, for future work, we plan to go beyond manual analysis and development, to an automated meta-analysis of a large number of comparable, published experiments. This will help increase the knowledge representation's scalability, robustness, and portability.

ASR FOR MOBILE DEVICES

When designers and developers build a speech interface, the job is only half done: it must then be deployed and monitored. To continuously improve the interface, developers need access to real data, which can only be collected by deploying an initial system. As mobile devices proliferate, many speech-recognition interfaces are now developed for such platforms. However, when compared with desktop ASR systems or those on central servers, implementation of accurate speech recognizers on mobile devices faces several challenges, including limited available storage space (language and acoustic models must be smaller, which leads to low performance), cheap and variable microphones that are often far from the speaker's mouth, low processing power without support of parallel processing (algorithms trade off speed and accuracy), and highly variable acoustic environments. Moreover, mobile devices consume a lot of energy during algorithm execution;[8] this is an important consideration for speech applications in the developing world, where electricity-supply issues could hamper the use of mobile applications in everyday settings.[9]

When compared with desktop ASR systems or those on central servers, implementation of accurate speech recognizers on mobile devices faces several challenges.

Before describing our proposed approach for these devices, we review two existing mobile ASR architectures that offer distinct advantages and disadvantages in light of these challenges. As alluded to earlier, the speech recognition process consists of two major steps: feature extraction, where the audio file is converted into features that accurately represent the acoustic information but take up much less memory than the raw audio; and ASR search, in which these features are used to identify the most likely text they might represent. Although feature extraction is a computationally intensive operation, it only consumes 2 percent of the processing time in the entire speech recognition process—98 percent is invested in search.[8]

Architecture 1: embedded mobile speech recognition

In the first architecture we describe, both processes involved in speech recognition—feature extraction and ASR search—happen locally on the mobile device.

Advantages. The main advantage of this mode is that it does not rely on any communication with a central server, and hence the applications that use such ASR can work in areas without any network connectivity, such as rural areas in developing regions or subway stations in developed ones. In other words, the ASR system is always "ready for use." Moreover, there is no cost and latency associated with transmitting and receiving information to and from the server, as in other modes. While monthly data plans can mitigate cost issues, network latency is a major issue in developing regions—especially the most rural—where cellular data connections are slow and unreliable, with frequent call drops. This mode also protects user privacy—no speech is transmitted to a central site.

Disadvantages. The disadvantage of this approach is that many mobile devices are not comparable to the high-end servers that can perform complex computations such as personalization algorithms for user or acoustic contexts. They fall somewhat short in terms of speed, runtime, and persistent memory, which restricts the type of applications that can be supported with this architecture. Also, no user speech is readily available for application developers to measure performance and iteratively improve the system.

Architecture 2: network speech recognition

In networked or cloud-based recognition, ASR search is shifted to a central server, with the mobile device sending the encoded audio to the server and getting back the recognition result. A slight variation of this approach is when the feature extraction is done locally to reduce the amount of information sent over the network, but the vast amount of computation goes to the server.

Advantages. This mode moves the burden of audio postprocessing as well as ASR search to a high-configuration server capable of executing real-time speech recognition systems. It can also utilize a much larger (and potentially more accurate) acoustic and language model, thereby offering significant advantages in terms of accuracy. Additionally, realistic user data is available to the application developer, making it much easier to improve the overall system. Most importantly, in the context of developing countries and regions, it can support speech applications on low-end mobile devices, such

as cell phones that are not capable of running a local ASR system.

Disadvantages. Despite increased resources in the form of a powerful central server, this mode has a number of drawbacks: it requires a continuous, reliable network connection to use the speech application; it loses acoustic information when the audio is encoded using low bit-rate codecs and packeted transmission; and the server needs to account for all variations in device, channel, speaker, or condition within a global set of parameters. Needless to say, this makes user-based adaptation difficult. Existing network-based ASRs from Apple and Google implement speaker-independent models, thereby compromising on accuracy for nontraditional cases, such as dialectal variations of a particular user group.

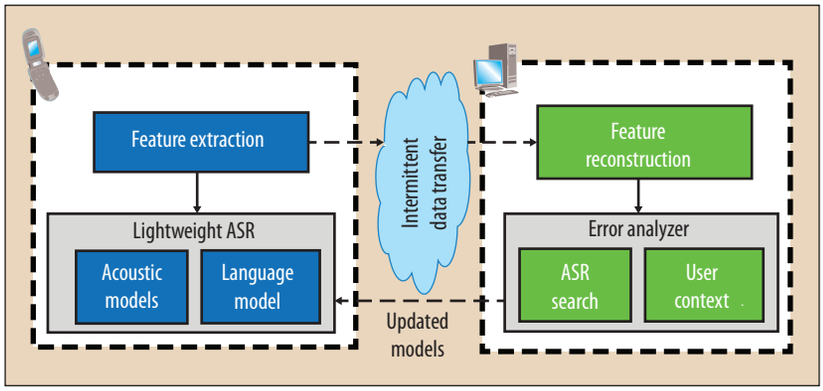

Figure 2. Hybrid approach. Automatic speech recognition (ASR) is done on the local mobile device, which benefits from user- and context-based personalization guided by intermittent server connection, whenever available.

Hybrid approach: decode locally, supervise remotely, then adapt

Our approach combines the other two modes—the major ASR subsystems (including feature extraction and ASR search) are on the mobile device, so it can perform recognition locally; and, as in Figure 2, applications do not break down under conditions of zero network connectivity. Whenever there is an intermittent cellular connection available, the mobile device sends the (stored) extracted feature vectors with metadata information about the user and device, such as noise levels, channel information, decoded outputs, and so on, to the server. This enables the server to evaluate the recognition performance for each user independently with a much larger vocabulary and acoustic database, to recommend user- and context-based adaptations, and to send updated models. For instance, a user accessing the application in a noisy background is likely to need noise-filtering adaptations in the acoustic models stored locally on the device. Similarly, a non-native speaker of a language is more likely to need adaptations to the local ASR's pronunciation dictionary. Hence, even though the server is not responsible for decoding the speech signal while the application is in use, the server's processing power is used for compute-intensive functions such as user- and context-specific error analysis, and adaptations using larger, shared resources.

We propose an architecture that combines the most important advantages of the two server-based approaches (high recognition accuracy, updatability, maintainability, and moderate mobile hardware requirements), with the advantage of local ASR, which is the ability to function without network connectivity. Centralized server power is used, not to perform online recognition but to ensure that over time recognition on local devices achieves high accuracy. The assumption, however, is that each mobile phone is being used by a few users in a few acoustic conditions, thereby enabling a reasonable user- and context-based adaptation. As an additional benefit, provisioning server capacity for average and not peak use is acceptable because adaptation does not need to be in real time.

However, we do not expect that industry leaders such as Google or Apple will implement this architecture, as their experience permits efforts in directly collecting datasets for minority languages without deploying an existing system.[10] We therefore expect to see highly accurate recognizers in languages of interest to these companies. However, many academic researchers and freelance application developers do not have the resources to collect data on the scale of Google and Apple. Our proposed architecture is of immense utility for them to easily and rapidly bootstrap an existing, high-quality recognizer for a new domain or user group.

Initial results

For the proposed architecture to be successful, it is important to understand the limitations of different speech recognizers when they run locally on a mobile device. This information would be useful during the development phase to pick the recognizer that best meets the application requirements.

To this end, we examined two mobile recognizers, pocketSphinx and SphinxTiny,[11] which are mobile versions of the popular, open source speech recognizer Sphinx. To our knowledge, these are the only two open source mobile recognizers scaled down for efficient performance on a wide number of mobile platforms. A few hacks can make other popular recognizers, such as Kaldi, work on select mobile devices as well. We used a Nokia N800 Internet tablet running Maemo Linux OS2008 with a TI OMAP 2420 ARM processor clocked at 330 MHz with 128 Mbytes of RAM, which is reflective of the kinds of devices that

most households in developing regions might own, say, five years from now.

Our results suggest that while it is possible to achieve real-time recognition using either of these recognizers,[11] a design choice must be made based on functionality requirements. We find pocketSphinx is superior if real-time recognition—minimizing delay—is key. It also works best for recognition tasks that have a fixed, predefined grammar or for small vocabulary tasks with fewer than 1,000 words. For open-ended, large vocabulary tasks such as dictation of emails or text messages, or tasks that might permit larger delays in exchange for better accuracy, SphinxTiny is a better choice. To take this further, we anticipate that with the support of a server—even under intermittent cellular connection—recognition accuracy can be drastically improved over time using online adaptation methods, or even the KB we described earlier.

The success of our proposed "decode locally, supervise remotely, then adapt" approach depends on the degree to which the local speech recognizer can be reconfigured and optimized, depending on the server's analysis. Given the numerous types of optimizations possible, the exact and most relevant choice depends on the use case—noisy backgrounds versus non-native speech, for example. Research is therefore needed to identify the type of optimizations that will be most beneficial to a user or device if only a certain amount of data can be transmitted over the network. Our architecture can adapt as the user's context changes over time. For instance, if a non-native speaker achieves native-speaker-like pronunciation, the server analysis can identify another metric such as noise, accounting for overall user history as well.

Therefore, some of the primary research questions we will explore in the future are as follows:

- How do we perform context- and user-specific error analysis on speech data received from a mobile device?
- How do we identify which adaptation techniques are best suited for a user's context?
- Without compromising accuracy, what type of tradeoffs do we perform to minimize the amount of data to be transmitted back to a mobile device?
- How do we determine the maximum size of updated models that can be transmitted from the server to the client?
- What steps do we take when more than one user accesses the device?

To address these questions, in addition to metadata such as caller ID and acoustic features, we plan to also send the recognition results (that is, hypothesis) from the mobile recognizer to the server. The server then independently analyzes the acoustic features using a much larger acoustic database as reference to compute a more accurate recognition hypothesis. Having both types of hypothesis has two advantages: first, it helps identify factors that affect recognition performance most in the user's context, and second, the more accurate hypothesis at the server can be used as a speech label to build and adapt future versions of the model before shipping back to the device.

As computing devices continue to shrink in size and proliferate to millions of users, speech input/output interfaces will also continue to grow in popularity. However, designing, developing, and deploying speech recognition application is still rife with major challenges. While it is possible to treat ASR as a commodity for mainstream American-accented English speakers, speech recognition research is likely to add the most value for non-English-language users, foreign-accented speakers of English, and nontraditional use-case scenarios. Unfortunately, there is no easy way to provide ASR functionality without a dedicated development effort, which is needed each and every time. This is unlikely to happen in practice.

Two developments will define state-of-the-art speech recognition over the next couple of years: the first is the dramatic improvement that can be achieved with deep learning techniques such as deep neural networks. Academia and industry have adopted them with breathtaking speed and they help improve ASR in particular for big data scenarios. The second is a push toward low-data or "zero resource" scenarios that enable speech recognition to be quickly developed for new languages using very little transcribed data, as is the goal in the Intelligence Advanced Research Project Activity (IARPA)-sponsored Babel research program (www.iarpa.gov/Programs/ia/Babel/babel.html). Both these developments bode well for the proposed approach: they facilitate providing better initial models, and they provide more techniques to adapt recognizers to specific use cases or scenarios.

We believe that speech input and output technologies can be fundamentally disruptive, enabling new research that will benefit low literacy users, children, the disabled, and so on. But even as it becomes easier for non-experts in core speech technologies to bootstrap a new speech recognizer for a specific scenario, constant monitoring is the key to deploying ASR broadly and successfully in a larger research context. We invite anyone to join us in our efforts to lower the barriers to entry or to simply use our systems, once fully developed, to create the next-generation speech-recognition interfaces.

References

1. G.P. Laput et al., "PixelTone: A Multimodal Interface for Image Editing," *Proc. SIGCHI Conf. Human Factors in Computing Systems* (CHI 13), ACM, 2013, pp. 2185–2194.

2. J. Thomason and D. Litman, "Differences in User Responses to a Wizard-of-Oz versus Automated System," *Proc. North American Chapter Assoc. Computational Linguistics: Human Language Technologies* (NAACL-HLT 13), 2013, pp. 796–801.
3. C. Abras et al., "User-Centered Design," *Encyclopedia of Human-Computer Interaction*, W. Bainbridge, ed., Sage Publications, 2004.
4. S.R. Klemmer et al., "SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces," *Proc. 13th Ann. ACM Symp. User Interface Software and Technology* (UIST 01), ACM, 2001, pp. 1–10.
5. T. Schultz et al., "SPICE: Web-Based Tools for Rapid Language Adaptation in Speech Processing Systems," *Proc. Interspeech*, Int'l Speech Communication Assoc., 2007; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.123.3902.
6. H. Soltau et al., "A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment," *Proc. Automatic Speech Recognition and Understanding Workshop* (ASRU 01), IEEE, 2001, pp. 214–217.
7. A. Kumar et al., "Formalizing Expert Knowledge for Developing Accurate Speech Recognizers," to appear in *Proc. Interspeech*, Int'l Speech Communication Assoc., 2013.
8. A. Schmitt, D. Zaykovskiy, and W. Minker, "Speech Recognition for Mobile Devices," *Int'l J. Speech Technology*, vol. 11, no. 2, 2008, pp. 63–72.
9. A. Kumar et al., "An Exploratory Study of Unsupervised Mobile Learning in Rural India," *Proc. SIGCHI Conf. Human Factors in Computing Systems* (CHI 10), ACM, 2010, pp. 743–752.
10. T. Hughes et al., "Building Transcribed Speech Corpora Quickly and Cheaply for Many Languages," *Proc. Interspeech*, Int'l Speech Communication Assoc., 2010, pp. 1914–1917.
11. A. Kumar et al., "Rethinking Speech Recognition on Mobile Devices," *Proc. Int'l User Interfaces for Developing Regions* (IUI4DR 11), ACM, 2011, pp. 10–15.

Anuj Kumar *is a PhD candidate in the Human-Computer Interaction Institute at Carnegie Mellon University. His research interests are in developing voice-user interfaces, applications of machine learning, and mobile computing. Kumar is a Siebel Fellow and a member of ACM and the International Society for Computers and Their Applications (ISCA). Contact him at anujkumar@cmu.edu.*

Florian Metze *is an assistant research professor at Carnegie Mellon University's Language Technologies Institute. His current research interests include low-resource speech recognition, multimedia analysis and summarization, and increasing the uptake of speech recognition in other disciplines. Metze received a PhD in computer science from Universität Karlsruhe (now Karlsruhe Institute of Technology), Germany. He is a member of ACM, IEEE, ISCA, and GI. Contact him at fmetze@cs.cmu.edu.*

Matthew Kam *is a senior researcher at the American Institutes for Research. He works on technology for broadening access to economic opportunities. Kam received a PhD in computer science from the University of California, Berkeley. Contact him at mkam@air.org.*