Standards & Emerging Technologies

Editor: Sumi Helal ■ University of Florida ■ helal@cise.ufl.edu

Pervasive Speech Recognition

Neal Alewine, Harvey Ruback, and Sabine Deligne

As mobile computing devices grow smaller and as in-car computing platforms become more common, we must augment traditional methods of human-computer interaction. Current methods are impractical and unsafe in the pervasive environment, so many device makers are turning to speech as an additional or preferred method of user interaction.

Although speech interfaces have existed for years, the constrained system resources of pervasive devices, such as limited memory and processing capabilities, present new challenges. Also, mobile environments are more demanding than traditional voice-response environments, given the higher background noise, poor microphone quality, and user's cognitive load while operating a pervasive device. Furthermore, despite these constraints, performance requirements call for systems that can accurately recognize complex dialogs and produce human-sounding, understandable speech output.

Previous issues of *IEEE Pervasive Computing* have addressed some of these challenges, discussing cyber foraging and its ability to offload computing resources from a constrained device[1] and presenting various techniques for disambiguating speech from ambient noise.[2] Here, we provide an overview of embedded *automatic speech recognition* on the pervasive device and discuss its ability to help us develop pervasive applications that meet today's marketplace needs.

AUTOMATIC SPEECH RECOGNITION

ASR recognizes spoken words and phrases. State-of-the-art ASR uses a *phoneme-based* approach for speech modeling: it gives each phoneme (or elementary speech sound) in the language under consideration a statistical representation expressing its acoustic properties. An ASR system's acoustic training consists of finely estimating the parameters associated with these phoneme-level representations using hundreds of hours of speech uttered by hundreds of speakers. These generic phone models are then articulated into word models to form the lexicon appropriate for the desired application. This approach offers the benefit of speaker independence and application flexibility. In ASR systems that are language dependent, the ASR systems' training process is largely automated and the same basic process is executed for each language.

Although phoneme-based ASR requires more device resources than simpler technologies, we can apply them to more complex tasks. Furthermore, because of the speaker independence, such systems can work out of the box for most applications. (The sidebar presents other, associated ASR technologies that improve overall system accuracy and provide additional functional elements required to create different types of applications.)

The quality of speech an ASR system receives depends largely on the microphone's type and location. Whenever manufacturing limitations allow, device makers prefer a directional, noise-reducing microphone positioned as close to the speaker as possible. ASR consists of transforming the raw speech input into a hypothesized sequence of words. As Figure 1 shows, three primary modules perform this process: the front-end, labeler, and decoder modules. Each contributes approximately the same computational workload, and they operate in succession.

In one instance of an ASR system, the front-end module inputs 16 kHz speech samples coded onto 16 bits. For every 15 ms of speech, it computes a 13-dimensional *mel-frequency cepstral coefficient* feature vector condensing the spectral information from an overlapping window containing 25 ms of speech. Unlike speech, noise sources such as car noise have significant energy in the low frequencies, so the system can discard these low-frequency signal components during MFCC computation. Finally, the front-end module performs energy normalization and mean normalization to reduce high variability in the signal levels and the effects of high variability of the audio input channel, respectively.

The labeler module appends first and second derivatives to the cepstral vectors and scores each of the resulting 39-dimensional vectors against the acoustic model, which consists of single-state Hidden-Markov Models. Each HMM specializes in modeling a given phoneme

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/04/$20.00 © 2004 IEEE

ASSOCIATED AUTOMATIC-SPEECH-RECOGNITION TECHNOLOGIES

Supervised adaptation, or *speaker enrollment*, improves accuracy for an individual speaker by having the user read a predetermined script. The system analyzes this specific audio and adapts the acoustic model to the user's specific characteristics. *Unsupervised adaptation* improves the accuracy for a specific user as well but doesn't require reading a predefined script. Instead, the system adapts during normal operation.

Name-tag or *voice-tag* support lets the user teach the system new words. Phone-dialing applications often use this technology to learn names in the address book. This feature is critical for applications requiring voice access to data that it can't predefine.

Dictation can recognize free-form commands. It lets the user speak in a more free-form style using statistical models as opposed to the finite-state grammars that most embedded systems currently use. The user is still bounded by the words and types of phrases that the designer used to train the system, but flexibility is allowed in the phrase structure, resulting in an easier to use system.

Natural language understanding helps systems interpret the meaning of free-form requests, because simple command-to-action mapping isn't sufficient. Approaches include keyword spotting, rule-based analysis, and statistical NLU. Statistical NLU employs models generated from domain-specific training sets representing how free-form phrases correlate to a predetermined set of actions and attribute-value pairs.

Dialog management builds on NLU by providing a method to interpret a phrase's meaning relative to the current user task and by controlling interaction with the user. The dialog manager applies the action and attribute-value pairs to a predefined set of user tasks and, when required, prompts the user for additional information.

Barge-in, *wake-up commands*, and *always listening* refer to technologies associated with how a user interacts with the system. Each removes the need for a user to press a button to speak. Barge-in refers to recognizing speech while an audio prompt is playing. Wake-up commands let the user speak a predefined phrase at any time, indicating that a command follows. Always listening is the final evolutionary step, letting the user speak any command at any time. Each approach requires echo cancellation to remove known audio signals, such as prompts or radio output, from the input signal. In one implementation, the speech recognition front-end module receives a separate channel containing only the known signal's reference samples, which are corrupting the user's speech. It then adaptively filters out this signal from the stream-of-speech samples.[1]

REFERENCE

1. E. Weinstein, M. Feder, and A.V. Oppenheim, "Multi-Channel Signal Separation by Decorrelation," *IEEE Trans. Speech and Audio Processing*, vol. 1, no. 4, 1993, pp. 405–413.

in a given phonetic context, called *allophone*. An example allophone would be the phoneme "a" occurring after a plosive (consonant sound where the flow of air is stopped and then released, such as "p") and before a nasal consonant (such as "n")—"pan." Each HMM state is associated with a Gaussian mixture letting us measure the probability that any given feature vector be an instance of the allophone associated to the HMM. When training the acoustic model, the number of allophones and the number of Gaussians allocated to each state are optimized to reach a fair tradeoff between memory and computational resources, and modeling accuracy.[3,4] The labeler establishes for each feature vector a ranked list of the few hundred states with the highest probabilities. A table lookup[3] then infers state likelihoods on the basis of their rank in the list and forwards the sequence of state likelihoods to the decoder.

The decoder module relies on allophone graphs that, given word transitions that describe a grammar or language model, specify the eligible sequences of allophones together with their a priori likelihood. In grammar-based applications or limited-domain dictation, allophone graphs are usually precompiled. The decoder reads the state likelihoods and implements a synchronous Viterbi search[5] to retrieve the most likely sequence of states among those permitted by the active allophone graph. A word identifier marking the end of each sequence of allophones modeling a word allows a fast mapping

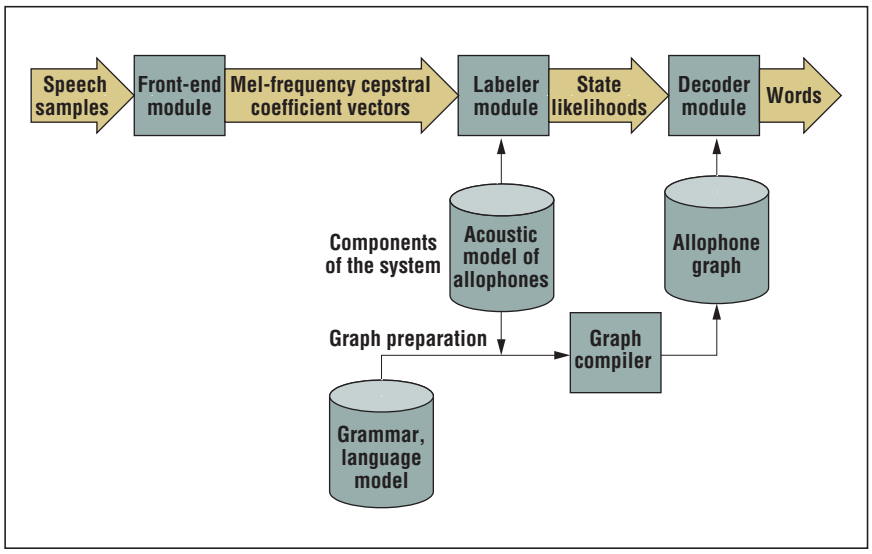


Figure 1. The main operations of speech recognition.

STANDARDS & EMERGING TECHNOLOGIES

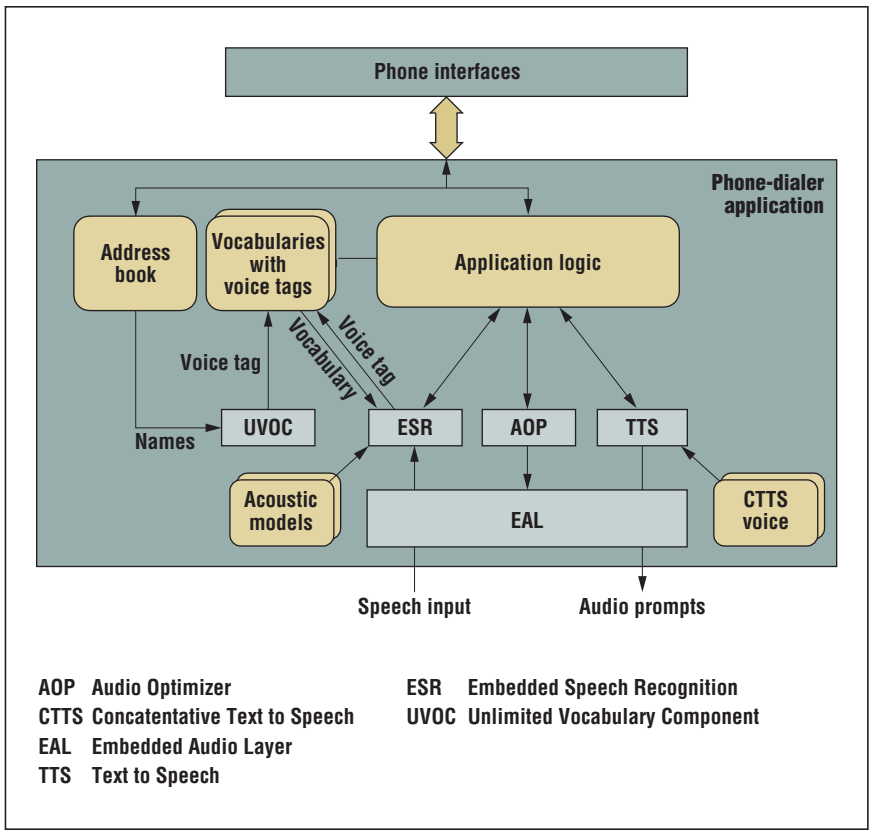


Figure 2. A phone-dialer application using IBM Embedded ViaVoice.

from the best sequence of states to the recognized words upon search completion. The result is a text-based best hypothesis that matches the initial raw-speech input.

SPEECH PROGRAMMING

A speech-programming interface's design and architecture is just as important to the application as the speech recognition engine's accuracy and quality. For a pervasive environment, interfaces must provide a portable, scalable, and highly flexible solution.

Portability lets application designers select a platform that meets their application's hardware and software requirements. Scalability lets them include in the final implementation only the required speech technology components. Data-driven speech components are also highly desirable, because they let designers modify the application behavior relative to the speech components through data-file updates rather than through code changes. The result is a more stable product that's readily customized and easily updated.

Speech user interface design and validation is a critical step during application development. Because people are accustomed to speaking to other people, designers must account for this preconditioned input method. Commands should be natural and intuitive, resulting in less memorization for new users. Also, acoustically distinct alternative commands for the same task ensure high accuracy for users with different speaking styles.

Speech user interface design is an iterative process, like most other design tasks, intended to account for the variability in user responses to system prompts. Speech recognition adds an additional complexity to user interface design because the interface's quality depends on the commands' accuracy in addition to their intuitiveness. The confusability of words is a major contributor to poor speech recognition. Acoustic confusability encompasses the phonetic similarities of pronunciations in the vocabulary, including how the background-noise environment amplifies the confusability of like-sounding words. So, an iterative design process that includes these real-world influences becomes even more valuable because it can demonstrate such usability issues early during design and development.

A PHONE-DIALER APPLICATION

IBM Embedded ViaVoice is a Software Developer's Kit that supports speech recognition, speech synthesis, and associated technologies. More important, it's an example of ASR technology that addresses many of the issues we've discussed. The SDK contains a set of scalable components that developers can use to meet specific application and platform requirements. EVV consists of the tools, data, and runtime components required to develop full-function speech recognition and text-to-speech applications in a variety of languages. Figure 2 shows a typical application, a hands-free phone-dialer that dials numbers based on speech commands. This application uses these EVV components:

- *Embedded Speech Recognition*. ESR is a speech recognition component that provides finite-state grammar support, voice-tag (word or phrase trained at runtime) processing, and all other recognition functionality.
- *Text-to-Speech*. The component that provides formant and concatenative synthesis.
- *Embedded Audio Layer*. EAL is an audio abstraction layer that provides applications with a platform-independent interface to basic audio functionality.
- *Audio Optimizer*. AOP provides automatic gain control and audio signal characteristic feedback features

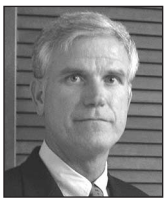
designed for speech recognition in dynamic noise environments.
- *Unlimited Vocabulary.* UVOC provides runtime grammar compilation and can generate pronunciations from spellings.

The phone-dialer application uses ESR for speech recognition and voice-tag support, ECI for prompts and command verification, AOP and EAL for audio management, and UVOC to generate pronunciation for names already existing in the address book. The EVV SDK also includes all the data components (acoustic models, text-to-speech voices, and dictionaries) and tools to manage this data. The EVV runtime components use the data blocks that the tools create to define application-specific behaviors such as the supported language, voice characteristics, and application vocabularies. The application in turn provides the interface logic tying these data blocks and runtime components together to produce the final application package and behavior. Using this data-driven model, an application can support multiple languages with the same implementation by providing the appropriate acoustic model, CTTS (Concatentative Text to Speech) voices, and vocabulary data files.

Future research and application of ASR in pervasive devices will focus on improved performance in high noise conditions, more natural and conversational speech interfaces, and improved personalization to better match specific user needs. Researchers are pursuing a range of technologies and techniques to address these needs including combined audio-visual processing, distributed speech recognition, and natural-language-understanding dialog-management systems. The successful application of these technologies in the pervasive environment will require a continued focus on system resource requirements, efficient speech programming techniques, and good speech user interface design. P

REFERENCES

1. R.K. Balan, "Powerful Change Part 2: Reducing the Power Demands of Mobile Devices," *IEEE Pervasive Computing*, vol. 3, no. 2, 2004, pp. 71–73.
2. T.E. Starner, "The Role of Speech Input in Wearable Computing," *IEEE Pervasive Computing*, vol. 1, no. 3, 2002, pp. 89–93.
3. L.R. Bahl et al., "Performance of the IBM Large Vocabulary Continuous Speech Recognition System on the ARPA *Wall Street Journal* Task," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP 95), IEEE Press vol. 1, 1995, pp. 41–44.
4. S.S. Chen and P.S. Gopalakrishnan, "Clustering via the Bayesian Information Criterion with Applications in Speech Recognition," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP 98), vol. 2, IEEE Press, 1998, pp. 645–648.
5. L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993, chap. 6.



Neal Alewine is a senior technical staff member at IBM Pervasive Computing. Contact him at alewine@us.ibm.com.

Harvey Ruback is a senior software engineer at IBM Pervasive Computing. Contact him at hruback@us.ibm.com.

Sabine Deligne is a research staff member at IBM Watson Research. Contact her at deligne@us.ibm.com.