petitive. It would be nice to design the error messages with different prompts according to recognition confidence measures and the dialog context.

Most current telephony systems do not have much intelligent dialog management beyond a simple state-based dialog model. Advanced dialog management as discussed in Chapter 17 is critical for error repairs, anaphora resolution, and context management. There are a number of institutions pursuing advanced telephony spoken language interfaces. You can find some of these excellent programs at Carnegie Mellon University[9] and Massachusetts Institute of Technology.[10]

18.2.3. Dictation

Dictation is attractive to many people, since speaking is generally faster than typing, especially in East Asian languages such as Chinese. There are a number of general-purpose dictation software products such as IBM's ViaVoice®, Dragon's NaturallySpeaking®, Lernout&Hauspie's Voice Xpress®, as well as the dictation capability built into Microsoft Office 2001. There are also vertical markets for dictation applications. Radiologists and lawyers are two examples of such niche markets. They are generally pressed for time. Radiologists are often hands- and eyes-busy when at work. Lawyers' work is often language intensive and they need to write long documents and reports. Both radiologists and lawyers have been using dictation devices, and they have a strong need to have customized language models for their specialized vocabulary. In fact, because of the constraints of the language they use, the perplexity is often much smaller than in the general business articles, leading to improved performance for both the medical and legal segments.

In dictation applications, it is important to convert speech into the text you would like to see in its written form. This implies that punctuation should be added automatically, dates and times should be converted into the form that is conventionally used, and appropriate capitalization and homonym disambiguation should all be seamless to users. Dictation is typically associated with word processing applications, which provide for a nice separation of input modes based upon the division of the primary task into subactivities:

- Text entry
- Command execution, such as *cross-out* to delete the previous word, or *capitalize-that*
- Direct manipulation activities such as cursor positioning and text selection

It is likely that separate single input modes for each activity can increase efficiency. In a study conducted by Morrison et al. [28], subjects switched modality from speech input to typed input while issuing text-editing commands and found the switch of modality *disruptive*. Karat et al. [19] compared the state-of-the-art dictation product with keyboard input.

[9] http://www.speech.cs.cmu.edu/speech/

[10] http://www.sls.lcs.mit.edu/sls/

They reported that several of the initial-use subjects commented that keyboard entry seemed *much more natural*. This reflects the degree to which some people have become used to using keyboards and the fact that speech recognition still makes too many mistakes. Experienced users can enter transcription text at an average rate of 107 uncorrected words per minute. However, correction took them over three times as long as entry time on average, leading to an average input rate nearly twice as slow as keyboard (including correction). Thus, unless the accuracy of speech recognition and its error correction can be significantly improved for the mass market, text entry is best performed by keyboard.

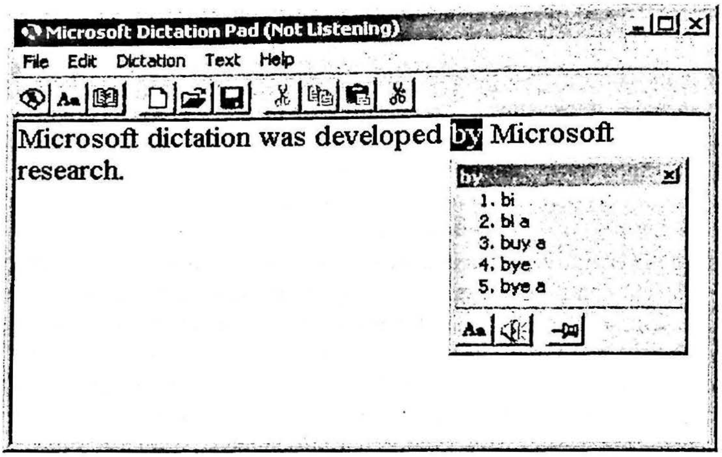
The mouse is generally accepted as being well suited to direct manipulation activities. Further, based on human-factors studies, full typed input of commands, single keyboard presses, and accelerator keys is not as efficient as speech-activated commands [26, 33]. The case for the utility of speech input in word-processing applications relies upon its superiority over the mouse with respect to the activation of commands. Word-processing and text-entry applications are also naturally *hands-busy, eyes-busy* applications. It is inconvenient and time consuming for the user to have to interrupt the typing of text or to move his/her eyes from the work in order to execute word-processing commands. This is not necessary with speech activation of commands.

To improve dictation throughput, you should make it as easy as possible for users to correct mistakes. As illustrated in Figure 18.3, you can provide easy access to the *Correction Window* so the user can correct mistakes that the recognizer made. Thus the quality of *n*-best or word-lattice decoding discussed in Chapter 13 is critical. We use examples drawn from Microsoft Dictation[11] to illustrate these concepts.

You should also allow the user to train the system so it can adapt to different accent and environment conditions. A typical training wizard, illustrated in Figure 18.4, asks users to read a number of sentences. It is important to use a confidence measure, as discussed in Chapter 9, to filter out unwanted training data so that you can eliminate possible divergences of training the acoustic model. In Figure 18.4, the user is prompted with the sentence shown in the window. As the user reads the sentence, the black progress bar covers the corresponding word. If the user misreads the word, the progress bar typically stays where it is.

Another practical problem for most dictation products is that they have about 60,000 to 100,000 words in the vocabulary, which contains all the inflected forms, so *work* and *works* are considered as two different words. As the vocabulary is limited (for detailed discussion on vocabulary selection, see Chapter 11), you should provide a capability to dynamically add new words to the system. This capability is illustrated in Figure 18.5. You can listen to the new word you added to see if the computer guessed the pronunciation of the word correctly or not. If you want to pronounce the word differently, you can either type in the words that sound the way you want it to or type in the phonetic sequence.

[11] Microsoft Dictation was developed by the authors at Microsoft Research, and was included for free in the SDK4.0.

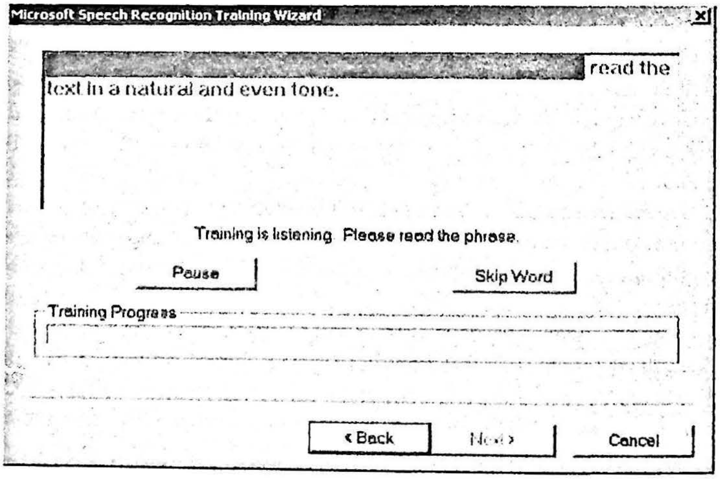


Microsoft dictation was developed by Microsoft research.

by

1. bi
2. bi a
3. buy a
4. bye
5. bye a

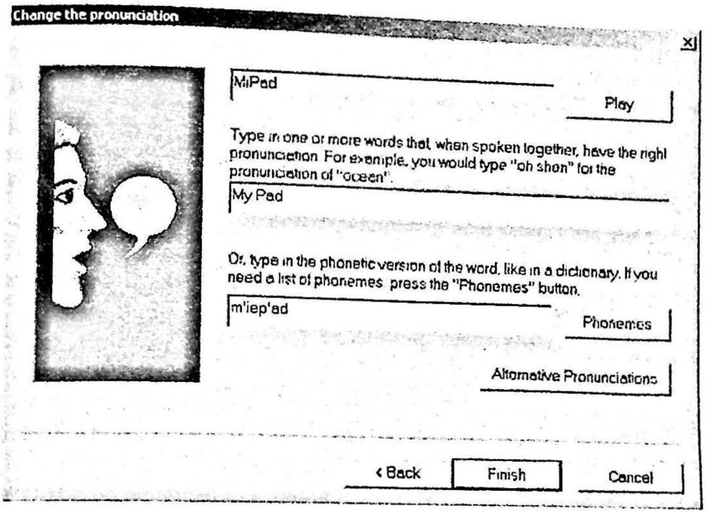
Figure 18.3 *Correction window* in Microsoft Dictation. By clicking the dictated word, the alternative words are listed in the correction window. If the dictated word is wrong, you can click the word in the correction window to easily replace the misrecognized word. Reprinted with permission from Microsoft Corporation.



read the text in a natural and even tone.

Training is listening. Please read the phrase.

Figure 18.4 Training wizard in Microsoft Dictation. Reprinted with permission from Microsoft Corporation.



Change the pronunciation

MiPad

Type in one or more words that, when spoken together, have the right pronunciation. For example, you would type "oh shen" for the pronunciation of "ocean".

My Pad

Or, type in the phonetic version of the word, like in a dictionary. If you need a list of phonemes, press the "Phonemes" button.

m'iep'ad

Figure 18.5 New word addition in Microsoft Dictation. Reprinted with permission from Microsoft Corporation.

18.2.4. Accessibility

Approximately 10–15 percent of the population has an impairment that affects their ability to use computers efficiently. Accessibility is about designing applications that everyone can use. The GUI actually makes accessibility a more critical issue. People with disabilities face more usability issues than those without. There are a number of Web sites dedicated to accessibility.[12]

People who cannot effectively use a keyboard or a mouse can use speech commands to control the computer and use the dictation software to input text. For example, Dragon's dictation software lets you control everything by voice, making it extremely useful for people with repetitive stress injury (RSI). A screen reader is a program that uses TTS to help people who have reading problems or are vision impaired. Windows* 2000 includes one such screen reader, called Narrator, which reads what is displayed on the screen: the contents of the active window, menu options, and text that the user types. Users can customize the way these screen elements are read. They can also configure Narrator to make the mouse pointer follow the active item on the screen and can adjust the speed, volume, and pitch of the narrator voice.

[12] http://www.el.net/CAT/index.html (The Center for Accessible Technology).
http://www.lighthouse.org/print_leg.htm (Lighthouse International).
http://www.w3c.org/wai (World Wide Web Consortium's Accessibility Page).
http://www.microsoft.com/enable (Microsoft Accessibility Home Page).

18.2.5. Handheld Devices

For handheld devices such as *smart phones* or *Personal Digital Assistants* (PDA), speech, pen, and display can be used effectively to improve usability. The small form-factor implies that most of the functions have to be hidden. Speech is particularly suitable to access information that can't be seen on the display. Current input methods are generally pen- or soft keyboard-based. They are slower than dictation-based text input. For form filling, speech is also particularly suitable to combine multiple parameters in a single phrase. The increasing popularity of cell phones is likely to be the major thrust for using handheld devices to access information.

Potential usability problems of handheld devices are the lack of privacy and the problem of environmental noise. It is hard to imagine that we can have 100 people in a conference room talking to their handheld devices simultaneously. Since these devices are also often used in a noisy environment, you need to have a very robust speech recognition system to deal with signals around 15 dB. In Section 18.5 we illustrate the process of designing and developing an effective speech interface and application for a handheld device.

18.2.6. Automobile Applications

People already use cell phones extensively when driving. Safety is an important issue whenever a driver takes his/her eyes off the road or his/her hands off the steering wheel. In many countries it is illegal to use a cellular phone while driving. Speech-based interaction is particularly suitable for this eyes-busy and hands-busy environment. However, it is a challenge to use speech recognition in the car due to the relative high noise environment. Environment robustness algorithms, discussed in Chapter 10, are necessary to make automobile applications compelling.

Figure 18.6 Clarion in-dash AutoPC.

One such system is Clarion's AutoPC, shown in Figure 18.6. It enables drivers to access information by speech while driving. You can tell your car what to do by speech. AutoPC has a vocabulary of more than 1200 words. It understands commands to change radio stations, provide turn-by-turn directions based on its Global Positioning System (GPS) navigation unit, or read e-mail subject lines using TTS. Running on Microsoft's Windows CE operating system, AutoPC's control module fits in a single dashboard slot, with the computer and six-disc CD changer stored in the trunk.

18.2.7. Speaker Recognition

The HMM-based speech recognition algorithms discussed in earlier chapters can be modified for speaker recognition, where a speaker model is needed for each speaker to be recognized. The term speaker recognition is an umbrella that includes speaker identification, speaker detection, and speaker verification. Speaker identification consists in deciding who the person is, what group the person is a member of, or that the person is unknown. Speaker detection consists in detecting if there is a speaker change. This is often needed in speech recognition, as a better speaker model can be used for improved accuracy. Speaker verification consists in verifying a person's claimed identify from his/her voice, which is also called speaker authentication, talker verification, or voice authentication.

In text-dependent approaches, the system knows the phrase, which can be either fixed or prompted to the user. The speaker speaks the required phrase to be identified or verified. The speaker typically has to enroll in the system by presenting a small number of speech samples to build the corresponding speaker model. Since the amount of training data is small, the parameter tying techniques discussed in Chapter 9 are very important.

Speaker recognition is a performance biometric, i.e., you perform a task to be recognized. It can be made fairly robust against noise and channel variations, ordinary human changes, and mimicry of humans and tape recorders. Campbell [3] and Furui [9] provided two excellent tutorial papers on recent advances in speaker recognition.

In general, accuracy is much higher for speaker recognition than for speech recognition. The best speaker recognition systems typically use the same algorithms, such as the ways to deal with environment noises and hidden Markov modeling, found in speech recognition systems. Applicable speaker recognition services include voice dialing, banking over a telephone network, telephone shopping, database access services, information and reservation services, forensic application, and voice login. There are a number of commercial speaker verification systems, including Apple's voice login for its iMac* and Sprint's Voice FONCARD*.

18.3. Speech Interface Design

No single unified interaction theory has emerged so far to guide user interface design. Study of the interaction has been largely associated with developing practical applications, having its roots in human factors and user studies. Practitioners have postulated general principles that we discuss in this section.

18.3.1. General Principles

There are a number of books on graphical user interface (GUI) design [10, 11, 17, 29, 41]. Usability research also generated a large body of data that is useful to UI designers. The Human Factors and Ergonomic Society (HFES) is developing the HFES-200 standard—

Ergonomics of Software User Interface, which has many insightful implications for interface design. The most important criteria, in order of importance, are:

- Effectiveness or usefulness
- Efficiency or usability
- User satisfaction or desirability

These three general points mean that, first and foremost, a user interface must work, so the user can get the job done. After that, it is important that the user interface accomplishes its task as productively as possible. Last, but not least, the user interface can be further improved by focusing on user satisfaction as measured by means of questionnaires and user feedback.

The major difference between GUI and spoken language interface is that speech recognition or understanding can never be perfect. Imagine you are designing for a GUI and your input method is a keyboard and a mouse where the error rate is about 5% for the key or mouse presses. These 5% errors make it much more difficult to have an effective user interface. In GUI applications, when you mistype or misclick, you regard these errors as your own mistakes. In spoken language applications, all the misrecognition errors are regarded as system errors. This psychological effort is a very important factor in the UI design. Thus it is not stretching to say that most of the design work for spoken language applications should be on how to effectively manage what to do when something goes wrong. This should be the most important design principle to remember.

We also want to stress the importance of iteratively testing the UI design. In practice, users, their tasks, and the work environment are often too varied to be able to have a good design without going through a detailed and thorough test program, especially for new areas like spoken language applications.

18.3.1.1. Human Limitations

Applications make constant demands on human capabilities. When we design the user interface, we must understand that humans have cognitive, auditory, visual, tactile, and motor limits. As a general principle, we should make the user interface easy to use, and not overload any of these limits.

For example, if the font is too small, the speech volume is too low, or the application requires people to constantly switch between a mouse and a keyboard, we have a poor interface, leading to lowered productivity. People have trouble remembering long complex messages spoken to them in a continuous fashion. If these messages are delivered with synthesized speech, the problem is worsened, because of the greater attention required to understand the synthesized speech. Effective user interface design requires a balance of easy access to information and an awareness of human limitations in processing this information. In general, people tend to handle well 7 (+ or – 2), pieces of information at a time.

18.3.1.2. User Accommodation

The interface and application should accommodate users as much as possible. This means that the design needs to match the way users want to do their work, not the other way around. People always have, or quickly create, a mental model of how the interface operates. You need to design and communicate a conceptual model that works well with the users' mental model. Their mental model should match the actual interface and the task, which is very important for the designer to consider consciously.

To accommodate users' needs, you must ask whether the interface you are designing helps people get their work done efficiently. Will it be easy to learn and use? Will it have an attractive and appropriate design? Will the interface fit individual tasks with a suitable modality? Will the interface allow the user to perceive that they are in control? Will the interface be flexible enough to allow the user to adjust the design for custom use? Does the interface have an appropriate tempo? Notice that speech is temporal. Does the interface contain sufficient user support if the user needs or requests it? Does the interface respond to the use in a timely fashion? Does the interface provide appropriate feedback to the user about the results of the actions and application status?

18.3.1.3. Modes of Interaction

The interface is unimodal if only one mode is used as both input and output modality. By contrast, multimodal systems use additional modalities in exchanging information with their users. When we interact with each other, we often rely on multimodal interaction, which is normal for us.

When speech is combined with other modalities of information presentation and exchange, it is generally more effective than a unimodal interface, especially for dealing with errors from speech recognition and understanding.

As an input modality, speech is generally not as precise as mouse or pen to perform position-related operations. Speech interaction can also be adversely affected by the ambient noise. When privacy is of concern, speech is also disadvantageous, since others can overhear the conversation. Despite these disadvantages, speech communication is not only natural but also provides a powerful complementary modality to enhance the existing keyboard- and mouse-based graphical user interface. As an input-output modality, speech has a number of unique characteristics in comparison with the traditional modalities:

- Speech does not require any screen real estate in the interface. For ubiquitous computing, this is particularly suitable for devices that are as small as our watches.
- Speech can be used at a distance, which makes it ideal for hands-busy and eyes-busy situations such as driving.

- Speech is expressive and very suitable to bring information that is hidden to the forefront. This is particularly suitable in the GUI environment or with handheld devices as the monitor space is always limited.
- Speech is omnidirectional and can communicate information to multiple users easily. However, this also has implications relating to privacy.
- Speech is temporal and sequential. Once uttered, auditory information is no longer available. This can place extra memory burden on the user and severely limit the ability to scan, review, and cross-reference information.
- For social interfaces, speech plays a very important role. For example, if we need to pose a question, make a remark, and delegate a task, there is no more natural way than speech itself.

Because of these unique features, we need to leverage the strengths and overcome the technology limitations that are associated with the speech modality. The conventional GUI relies on the visual display of objects of interest, the selection by pointing, and continuous feedback. Direct manipulation is generally inadequate for supporting fundamental transactions in applications such as word processing and database queries, as there are only limited means to identify objects, not to mention that ambiguity in the meanings of icons and limitations in screen display space further complicate the usability of the interface.

Generally, multimodality can dramatically enhance the usability of speech, because GUI and speech have complementary strengths and weaknesses. The strengths of one can be used to offset the weaknesses of others. For example, pen and speech can be complementary and they can be used very effectively for handheld devices. You can use *Tap and Talk* as discussed in Section 18.4 to activate microphone and select appropriate context for speech recognition. The respective strengths and weaknesses are summarized in Table 18.2. The advantage of pen is typically the weakness of speech and vice versa. This implies that user interface performance and acceptance could increase by combining both. Thus, visible, limited, and simple actions can be enhanced by nonvisible, unlimited, and complex actions. The multimodal interface is particularly suitable to people for completing contrastive functions or when task attributes are perceived as separable. For example, speech and pen can be used in different ways to designate a shift in context or functionality. Although most people may not prefer to use speech and pen simultaneously, they like to use speech to enter data and pen for corrections. Other contrastive tasks may include data versus commands and digits versus text. When inputs are perceived as integral, unimodal interfaces are efficient and work best.

Table 18.2 Complementary strengths of pen and speech for multimodal user interface.

Pen	Speech
Direct manipulation, requires hands and eyes	Hands/eyes-free manipulation
Good at performing simple actions	Good at performing complex actions
Visual feedback	Visual and location independent references
No reference ambiguity on the display	Multiple ways to refer to objects

18.3.1.4. Technology Considerations

Speech recognition applications can be divided into many classes, as illustrated in Figure 18.7. The easiest category is the system that can only deal with isolated speech with limited vocabulary for a command-and-control application, where typically a context-free grammar is used to constrain the search space, resulting in a limited number of ways users can issue a command. The hardest is the one that can support channel- and speaker-independent continuous speech recognition with both dictation and robust application-specific understanding capability. This is also the one that can really add value to a wide range of speech applications. In Figure 18.7, a *command-and-control* application implies application-specific restrictive understanding capability—typically with less than 1000-word vocabulary and with a lower perplexity context-free grammar. A *dictation* application implies general-purpose dictation capability—typically with more than 60,000 words in the lexicon and a stochastic *n*-gram language model. A *dictation and SLU* application implies both general-purpose dictation and less restrictive application-specific spoken language understanding capability—typically with 60,000 words in the lexicon with both *n*-gram and semantic grammar language models for both recognition and understanding.

Although a number of parameters can be used to characterize the capability of speech recognition systems, user-centered design often dictates that you need to use the most challenging technology solutions. The speaking style (from isolated speech to continuous speech) can dramatically change the performance of a speech recognition system. Natural, spontaneous, continuous speech remains the most difficult to recognize; but it is more user friendly. Small vocabulary (less than 50 words) is generally more accurate than large vocabulary (greater than 20,000 words); but the small vocabulary limits the options the user has on the task. The lower-perplexity language model generally has a great recognition accuracy; but it imposes a considerable constraint on how the user can issue a command. Most systems may work in the high-SNR (>30 dB) environment; but it is desirable that the system should still work when the SNR is below 10 dB.

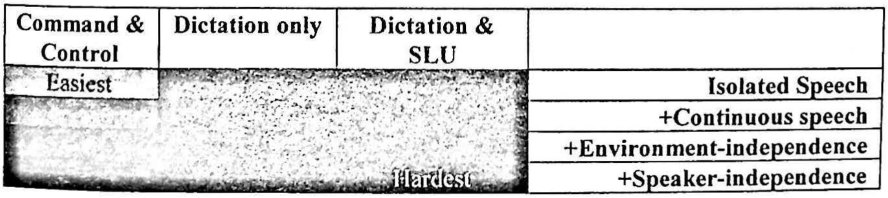


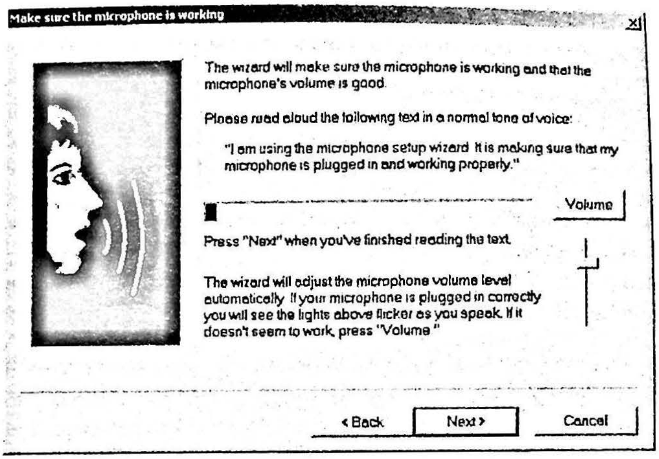
Figure 18.7 Illustrative degrees of difficulty for different classes of speech recognition (gray level from easy to hard).

Associated with the input modality is the speech output modality for many speech applications. Speech as an output medium taxes user memory and may even cause loss of context and distraction from the task if it is too verbose. Most of the text-to-speech systems have a problem generating a prosody pattern that is dynamic and accurate enough to convey the underlying semantic message. Recorded speech still has the best quality; but it is generally inflexible in comparison to TTS. There are many ways to provide additional information to the TTS so that the final synthesized quality can be improved dramatically. Examples include the use of recorded speech for constant messages, transplanted prosody, and concept-to-speech synthesis as discussed in Chapters 16 and 17.

Improved quality and ease of use often need expensive authoring. This is true for speech recognition, synthesis, and understanding. For example, to support channel independence, you need not only additional channel normalization algorithms to deal with both additive and convolutional noises but also realistic training data to estimate model parameters. To support robust understanding, you need to author semantic grammars that have broad coverage. This often requires tedious authoring and labor-intensive data collection and annotation to augment the grammar. Ultimately, you need a modular and reusable system that can warrant the development costs. To make effective use of dialog and error-repair strategies, you need to adopt tools for preventing dialog design problems during the early stages, ones that guide the choice of words in dialog and feedback and ensure usability and correctness in the context, etc. In addition, as discussed in Chapter 17, whether you have a system-initiative dialog, user-initiative dialog, or mixed-initiative dialog will have significant ramifications on the authoring cost.

A spoken language application requires certain hardware and software on the user's computer in order to run.[13] Most new sound cards work well for speech recognition and text-to-speech, including Sound Blaster™. Some old sound cards are only *half duplex* (as opposed to *full duplex*). If a sound card is half duplex, it cannot record and play audio at the same time. Thus it cannot be used for barge-in and echo cancellation. The user can choose among different kinds of microphones: a close-talk headset microphone that is held close to the mouth, a medium-distance microphone, or a microphone array device that rests on the computer 30 to 60 centimeters away from the speaker. A headset microphone is often needed for noisy environments, although microphone array or blind separation techniques have the potential to close the gap in the future (see Chapter 10). Most speech products also need to calibrate the microphone gain by speaking one or two utterances, as illustrated in Figure 18.8. It adjusts the gain of the amplifier to make sure there is an appropriate gain without clipping the signal. This can be done in the background without distracting the user.

[13] Speech recognition engines currently on the market typically require a Pentium 200 or faster processor. Speech recognition for command and control consumes 1 to 4 MB RAM, and dictation requires an additional 16-32 MB. Text-to-speech engines use about 1-3 MB, but high-quality TTS can require more than 64 MB.



The wizard will make sure the microphone is working and that the microphone's volume is good.

Please read aloud the following text in a normal tone of voice:

"I am using the microphone setup wizard. It is making sure that my microphone is plugged in and working properly."

Press "Next" when you've finished reading the text.

The wizard will adjust the microphone volume level automatically. If your microphone is plugged in correctly you will see the lights above flicker as you speak. If it doesn't seem to work, press "Volume."

Figure 18.8 Microphone setup wizard used in Microsoft Speech SDK 4.0. Reprinted with permission from Microsoft Corporation.

18.3.2. Handling Errors

Speech recognition and understanding can never be perfect. Current spoken language processing work is very much a matter of minimizing errors. Together with the underlying speech-engine technologies, the role of the interface is to help reduce the errors or minimize their severity and consequences. As discussed in earlier chapters, there are errors associated with word deletion, word insertion, and word substitution. The recognizer may be unable to reject noises and classify them as regular words. The speaker may also have new words that are not in the lexicon, or may have less clear articulation due to accent, stress, sickness, and excitement.

To repair errors in the system, we need to take each individual application into consideration, since the strategy is often application-dependent. To have a compelling application, we must design the application with both the limitation of the engine and the interaction of the engine and UI in mind. For example, multimodal applications are generally more compelling and useful than speech-only solutions as far as error handling is concerned. A multimodal application can overcome many limitations of today's SLU technologies, but it may be inappropriate for hands-busy or eyes-busy scenarios.

18.3.2.1. Error Detection and Correction

We need to detect errors before we can repair them. This is a particularly difficult problem that often requires high-level context and discourse information to decipher whether there are errors. We can certainly use confidence measures, as discussed in Chapter 9, to detect potential errors, although confidence measures alone are not particularly reliable.

There are many ways to correct the errors. The choice of which one to use depends on the types of the applications you are developing. We want to emphasize that correction based on another modality is often desirable, since it is unusual that two independent modalities make the same mistake. For example, speech recognition errors are often different from handwriting recognition errors. In addition to the complementary nature of these different errors, we can also leverage other modalities for improved error-correction efficacy. For example, a pen is particularly suitable for pointing, and we can use that to locate misrecognized words. A camera can be used to track whether the speaker is speaking in order to eliminate background noise generated from other sound sources.

Let's consider a concrete example. If you have a handheld device, you may be able to see the errors and respeak the desired commands or use a different modality such as pen to help task completion. When you want to create an email using speech, and a dictation error occurs, you can click the misrecognized word or phrases to display a list of *n*-best alternatives. You can select the correct word or phrase from the *n*-best list with the pen. Alternatively, you can respeak the highlighted word or phrase. Since the correct answer should not be the previous misrecognized word or phrase, you can exclude them in the second trial. If it is an isolated word in the correction stage, you can also use the correct silence context triphone models for the word boundary as the correct acoustic model. You can also use the correct *n*-gram language model from the left context with the assumption that the left words are all correct. With these correct acoustic and language models, it is expected that the second trial would have a better chance to get the right answer.

If you have telephony-based applications, you can count on keywords such as *no*, *I meant* to identify possible recognition or understanding errors. When errors occur, you should provide specific error-handling messages so that the user knows what response the system wants. An adequate error message should tell the user what is wrong, how to correct it, and where to get help if needed. Let's consider an example of the following directory assistant application:

Computer: Please speak the name

User: Mike

Computer: Please speak the name

User: Mike

Clearly, the user does not know what responses the computer wants. An improved dialog can be:

Computer: Please speak the first and last name

User: Mike Miller

Computer: Which division is Mike Miller working?

User: Research group

You can use alternative guesses intelligently for telephony applications, too. For example, if the user says "*Mike Miller*" and the computer is unsure of what was said, it can prompt the user to repeat with a constrained grammar. In the example above, if the computer is unsure whether the user said Mike or Mark, it may want to respond with "*Did you say Mike Miller or Mark Miller*?" before it proceeds to clarify which division Miller is working in.

When an error is detected, the system should be adapted to learn from the error. The best error-repair strategy is to use an adaptive acoustic and language model so that the system can learn from the errors and not repeat the same mistake again. This is why adaptive acoustic and language modeling, as discussed in Chapters 9 and 11, is critical for real acceptance of speech applications. When an error occurs and the computer understands the error via user correction, you can use the corrected words or phrases to adapt related components in the system.

18.3.2.2. Feedback and Confirmation

Users need feedback from the system so that they understand the status of the interaction. When a command is issued, the user expects the system to acknowledge that the command is heard. It is also important to tell the users if the system is busy so that they know when to wait. Otherwise, the users may interpret the lack of response or feedback as errors of recognition or understanding.

If the user has requested that an action be taken, the best response is to carry out the action as requested, but if the action cannot be carried out, the user must be notified. Sometimes, the user may need this feedback to know whether it is her turn to speak again. If the interface is multimodal, you can use visual feedback to indicate the status of the interaction. For example, in Microsoft's MiPad research prototype, when the microphone is activated with a pen pointing to a specific field, the microphone levels are indicated in the feedback rectangle in strip-chart fashion as shown in Figure 18.9 for the corresponding field. Initially this wipes from left to right, but when the right edge is reached, the contents scroll to the left as each new sample is added. The height of each sample indicates the audio level; there is some indication given for the redline level for a possible saturation. It also shows the total number of frames as well as the number of remaining frames to be processed. Then, as the

Figure 18.9 MiPad's dynamic volume meter. Reprinted with permission from Microsoft Corporation.

recognizer processes, the black bar wipes from left to right to indicate its progress. When the recognizer completes (or is canceled by the user), the dynamic volume meter is removed. The volume meter feedback described addresses a number of issues that are critical to most users: if the computer is listening or not; if the volume is too high or not, and when the computer finishes processing the speech data.

If the interface has no visual component, you can use auditory cues or speech feedback. If there is more than a 3-second delay after the user issues a command and the system responds, an auditory icon during the delay, such as music, may be used. A delay of less than 2-3 seconds is acceptable without using any feedback.

Since there are errors in spoken language systems, you may have to use confirmation questions to assure that the computer heard the correct message. You can associate risks with different commands, so you can have a different level of strategy for confirmation. For example, *format disk* should have a higher threshold or may not be allowed by voice at all. You can have an explicit confirmation before executing the command. You can also have implicit confirmation based on the level of the risks in the corresponding confidence scores from the recognizer. For example, if the user's response has more than one possible meaning (X and Y) or the computer is not confident about the recognized result (X or Y), the computer could ask, "*Do you want to do X or Y?*" You want to be specific about what the system needs. A prompt like "*Do you mean X or Y?*" is always better than *Please repeat*. You have to balance the cost of making an error with the extra time and annoyance in requiring the user to confirm a number of statements.

In a telephony-based application, you need to tell the users specifically that they are talking to a computer, not a real person. When AT&T used a long greeting for its telephone customer service, their prompts were as follows:

"AT&T Automated Customer Service. This service listens to your speech and sends your call to the appropriate operator. How may I help you?"

The longer version resulted in shorter utterances from the people calling in. The short automated version did not help shorten the utterances, because people did not seem to catch the automated connotation.

You can also design the prompt in such a way that you can constrain how people respond or use their words. Let's consider an example of the following directory assistant application:

Computer: Welcome to Directory Assistant. We look forward to serving you. How can I help you?

User: I am interested in knowing if you can tell me the office number of Mr. Derek Smith?

To have the user follow the lead of the computer, an improved dialog can be:

Computer: Welcome to Directory Assistant. Please say the full name of the person.

User: Derek Smith

When the system expects the user to respond with both the first name and last name, you should still design the grammar so that the system can recognize a partial name with lower probabilities. This makes the system far more robust—a good example of combining engine technology and interface seamlessly for improved user satisfaction.

18.3.3. Other Considerations

When speech is used as a modality, you should remember the following general principles:

18.3.3.1. Don't Use Speech as an Add-On Feature

It's poor design to just bolt speech recognition onto an application that is designed for a mouse and keyboard. Such applications get little benefit from speech recognition. Speech recognition is not a replacement for the keyboard and mouse, but in some circumstances it is a better input device than both. Speech recognition makes a terrible pointing device, just as the mouse makes a terrible text entry device, or the keyboard is bad for drawing. When speech recognition systems were first bolted onto the PC, it was thought that speaking menu names would be really useful. As it turns out, very few users use speech recognition to access a window menu.

18.3.3.2. Give Users Options

Every feature in an application should be accessible from all input devices: keyboard, mouse, and speech recognition. Users naturally use whichever input mechanism provides them the quickest or easiest access to the feature. The ideal input device for a given feature may vary from user to user.

18.3.3.3. Respect Technology Limitations

There are a number of examples of poor use of speech recognition. Having a user spell out words is a bad idea for most recognizers, because they are too inaccurate unless you constrain the recognizer for spelling. An engine generally has a hard time detecting multiple speakers talking over each other in the same digital-audio stream. This means that a dictation system used to transcribe a meeting will not perform accurately during times when two or more people are talking at once. An engine cannot hear a new word and guess its spelling unless the word is specified in the vocabulary. Speakers with accents or those speaking in nonstandard dialects can expect more errors until they train the engine to recognize their speech. Even then, the engine accuracy will not be as high as it would be for someone with the expected accent or dialect. An engine can be designed to recognize different accents or dialects, but this requires almost as much effort as porting the engine to a new language.

18.3.3.4. Manage User Expectations

When you design a speech recognition application, it is important to communicate to the user that your application is speech aware and to provide him or her with the commands it understands. It is also important to provide command sets that are consistent and complete. When users hear that they can speak to their computers, they instantly think of *Star Trek* and *2001: A Space Odyssey*, expecting that the computer will correctly transcribe every word that they speak, understand it, and then act upon it in an intelligent manner. You should also convey as clearly as possible exactly what an application can and cannot do and emphasize that the user should speak clearly, using words the application understands.

18.3.4. Dialog Flow

Dialog flow is as important to speech interface as screen design is to GUI. A robust dialog system is a prerequisite to the success of the speech interface. The dialog flow design and the prompting strategy can dramatically improve user experience and reduce the task complexity of the spoken language system.

18.3.4.1. Spoken Menus

Many speech-only systems rely on tree-structure spoken menus as the main navigation vehicle when no sophisticated dialog system is used. The design of these menus is, therefore, critical to the usability of the system. If there are only two options, you can have "*Say 1 for yes, or 2 for no.*" However, if a menu has more than two options, you should describe the result before you specify the menu option. For example, you want to have the system say "*For email say 1. For address book say 2…*" instead of "*Say 1 for email. Say 2 for address book….*" This is because the user may forget which choice he wants before hearing all the menu options.

You should not use more than five options at each level. If there are more than five, you should consider submenus. It is important to place the most frequently used options at the top of the menu hierarchy.

You should also design the menu so that commands sound different. As a rule of thumb, the more different phonemes you have between two commands, the more different they sound to the computer. The two commands, *go* and *no,* only differ by one phoneme, so when the user says, "*Go*" the computer is likely to recognize "*No.*" However, if the commands were "*Go there*" and "*No way*" instead, recognition would be much easier.

18.3.4.2. Prompting Strategy

When you design the prompt, use as few words as possible. You should also avoid using a personal pronoun when asking the user to respond to a question. This gives the user less to remember, as speech output is slow and one-dimensional. Typical users are able to remember three to four menu or prompt options at a time. You should also allow users to interrupt the computer with *barge-in* techniques, as discussed in Chapter 10.

It is often a good idea to use small steps to query the user at each step if application has a speech-only interface progressively. You can start with short high-level prompts such as "*How can I help you?*" If the user does not respond appropriately, the system can provide a more detailed prompt such as "*You can check your email, calendar, address book, and your home page.*" If the user still does not respond correctly, you can give a more detailed response to tell the user how to speak the appropriate commands that the computer can understand.

If the application contains users' personal information, you may want to treat novice users and experienced users differently. Since the novice users are unfamiliar with the system and dialog flow, you can use more detailed instructions and then reduce the number of words used the next time when the user is going through the same scenario.

18.3.4.3. Prosody

For TTS, prosodic features, including pitch, volume, speed, and pause, are very important for the overall user experience. You can vary pitch and volume to introduce new topics and emphasize important sections. Increased pitch dynamic range also makes the system sound lively.

Users tend to mimic the speed of the system. If the system is reacting and speaking slowly, the user may tend to slow down. If the computer is speaking quickly, the user may speed up. An appropriate speech rate for natural English speech is about 150 to 170 words per minute.

Pauses are important in conversation, but they can be ambiguous, too. We generally use pauses to suggest that it is the user's turn to act. On the other hand, if the system has requested a response from the user and the user has not responded, you should use the timeout period (typically 10 seconds) to take appropriate actions, which may be a repeat of a more detailed prompt or may be a connection to an operator. People are generally uncomfortable with long pauses in conversation. If there are long pauses in the dialog flow, people tend to talk more and use less meaningful words, resulting in more potential errors.

18.4. INTERNATIONALIZATION

To meet the demands of international markets you need to support a number of languages. For example, Microsoft Windows 2000 supports more than 40 languages. Software internationalization typically involves both globalization and localization phases. Globalization is the process of defining and developing a product such that its core features and code design do not make assumptions about a single language. To adapt the application for a specific market, you need to modify the interface, resize the dialog boxes, and translate text to a specific market. This process is called localization. The globalization phase of software internationalization typically focuses on the design and development of the product. The product designers plan for locale-neutral features to support multicultural conventions such as address formats and monetary units. Development of the source code is typically based on a single code base to have the capability to turn locale-specific features on and off without modifying the source code, and the ability to correctly process different character encoding methods. Products developed with the single worldwide binary model typically include complete global functionality, such as support for Input Method Editors used in Asian languages and bidirectional support for Arabic and Hebrew languages.

In addition to the most noticeable change in the translated UI, spoken language engines such as speech recognition or text-to-speech synthesis require significant undertakings. You need to have not only the lexicon defined for the specific language but also a large amount of speech and text data to build your acoustic and language model. There are many exceptional rules for different languages. Let's take dictation application as an example. When you have inverse text normalization, whole numbers may be grouped differently from country to country, as shown in Table 18.3. In the United States, we group three digits separated by a comma. However, many European locales use a period as the number separator, and Chinese conventions typically do not group numbers at all.

Table 18.3 Examples of how numbers are grouped by different locales for inverse text normalization.

123,456,789.0	United States and most locales
123456789.0	Chinese
12,34,56,789,0	Hindi

For the speech recognition engine localization, there are a number of excellent studies on both Asian [5, 15, 16, 21, 24, 32, 34, 37] and European languages [7, 8, 20, 22, 27, 31, 40]. Many spoken language processing components, such as the speech signal processing module, the speech decoder, and the parser, are language independent. Major changes are in the *content* of the engine, such as the lexicon and its associated processing algorithm, the grammar, and the acoustic model or language model parameter.

In general, it is easier to localize speech recognition than either text-to-speech or spoken language understanding. This is because a speech recognition engine can be mostly automatically derived from a large amount of speech and text data, as discussed in Chapters 9 through 13. For Chinese, you need to have special processing for tones, and the signal-processing component needs to be modified accordingly [4]. For both Chinese and Japanese the lexicon also needs to be carefully selected, since there is no space between words in the lexicon [12, 46]. As discussed in Chapter 17, the parser can be used without much modification for SLU systems. Most of localization efforts would be on the semantic/syntactic grammar and dialog design, which is language specific by nature.

As discussed in Chapters 14 and 15, the TTS front end and symbolic prosody components are much harder to localize than the TTS back end. The TTS back end can be easily localized if you use the trainable approach discussed in Chapter 16 on the assumption that the speech recognizer for the language is available. The internationalization process mainly consists of creating tools that can generate training phrase lists from target language texts and dictionaries, and ensuring that any reasonable-sized set of symbolic distinctive language sound inventories (phone sets) can be accepted by the system. The human vocal tract is largely invariant across language communities, and this relative invariance is reflected in the universality of the speech synthesis tools and methods.

When the lexical, syntactic, and semantic content is localized, you should be aware that the incorrect use of sensitive terms can lead to outright product bans in some countries, while other misuses are offensive to the local culture. For example, Turkey forbids the use of the term *Kurdistan* in text and any association of Kurdistan with Turkey. Knowing proper terminology is critical to the success of your application internationalization.

18.5. Case Study—MiPad

An effective speech interface design requires a rigorous process that typically consists of three steps with a number of iterations based on the user evaluation:

- Specifying the application
- Rapid prototyping
- Evaluation

We use Microsoft's research prototype MiPad [18] as an example to illustrate how you can use the process to create an effective speech application. As a wireless Personal Digital Assistant (PDA), MiPad fully integrates continuous speech recognition (CSR) and spoken language understanding (SLU) to enable users to accomplish many common tasks using a multimodal interface and wireless technologies. It tries to solve the problem of pecking with tiny styluses or typing on minuscule keyboards in today's PDAs or smart phones. It also avoids the problem of being a cellular telephone that depends on speech-only interaction. MiPad incorporates a built-in microphone that activates whenever a field is selected. As a user taps the screen or uses a built-in roller to navigate, the tapping action narrows the number of possible instructions for spoken language processing. MiPad runs on a Windows CE Pocket PC with a Windows 2000 Server where speech recognition is performed. The CSR engine has a 64,000 word vocabulary with a unified context-free grammar and *n*-gram language model as discussed in Chapter 11. The SLU engine is based on a robust chart parser and a plan-based dialog manager discussed in Chapter 17.

18.5.1. Specifying the Application

The first step to develop an application is to identify key features that can help users. You need to understand what work has been done, and how that work can be used or modified. You need to ask yourself why you need to use speech for the applications and review project goals and time frames. The first deliverable will be the spec of the application that includes a documented interface design and usability engineering project plan. Some of the key components should include:

- Define usability goal, not technology
- Identify the user groups and characteristics, interview potential users, and conduct focus-group studies
- Create system technology feasibility reports
- Specify system requirements and application architecture
- Develop development plan, QA test plan, and marketing plan

You should develop task scenarios that adopt a user's point of view. Scenarios should provide detail for each user task and be in a prose or dialog format. You may also want to include frequency information as a percentage to illustrate whether a particular case is happening more than 20% or 80% of the time, which helps you focus on the most common features first.

To create a conceptual interaction model, not the software architecture, we need to have a mental model for the users. Users have to manipulate interface objects either literally or figuratively in the navigation. Interfaces without clear and consistent user objects are difficult to use. Generally, you need a script flow diagram if speech is used. This describes how the user moves through dialogs with the system. To create a script flow diagram from your scenarios, you can start at the beginning of a scenario and derive a set of possible dialogs the user needs to have with the system to complete the scenario. You can draw a box representing a dialog with arrows that shows the order and relationship among the dialogs. If your interface involves graphic display, you need to include window flow diagrams as well, which organize objects and views and define how the user moves from one object to another. Of course, these two diagrams may be combined for effective multimodal interaction.

When the initial spec is in place, you can use the conceptual model to create actual paper windows and dialogs for the interface. You must iterate the design several times and get feedback early. Paper prototyping allows you to do these tasks most efficiently. It is useful to conduct walkthroughs with a number of people as though they were performing the task. You may want to start usability testing to collect real data on real users before the software is implemented. Even a system that is well designed and follows the best proactive approach may reveal some usability problems during testing. Usability testing has established protocols, procedures, and methodologies to get reliable data for design feedback [36, 42]. Generally you need to decide what to test, create usability specifications for the tasks, select the right user groups, specify exact testing scenarios, conduct the actual tests with participants, analyze the data, and report critical findings.

How did we do this for MiPad? Since MiPad is a small handheld device, the present pen-based methods for getting text into a PDA (Graffiti, Jot, soft keyboard) are barriers to broad market acceptance. As an input modality, speech is generally not as precise as mouse or pen to perform position-related operations. Speech interaction can be adversely affected by the ambient noise. When privacy is of concern, speech is also disadvantageous, since others can overhear the conversation. Despite these disadvantages, speech communication is not only natural but also provides a powerful complementary modality to enhance the pen-based interface. Because of these unique features, we need to leverage the strengths and overcome the technology limitations that are associated with the speech modality. Pen and speech can be complementary and they can be used very effectively for handheld devices. You can tap to activate a microphone and select appropriate context for speech recognition. The advantage of pen is typically the weakness of speech and vice versa. This implies that user interface performance and acceptance could increase by combining both. Thus, visible, limited, and simple actions can be enhanced by nonvisible, unlimited, and complex actions.

Table 18.4 Benefits of having speech and pen for MiPad.

Action	Benefit
Ed uses MiPad to read an email, which reminds him to schedule a meeting. Ed taps to activate microphone and says *Meet with Peter on Friday.*	Using speech, information can be accessed directly, even if not visible. Tap and Talk also provides increased reliability for speech detection.
The screen shows a new appointment to meet with Peter at 10:00 on Friday for an hour.	An action and multiple parameters can be specified in only a few words.
Ed taps Time field and says *Noon to one thirty.*	Field values can be easily changed using field-specific language and semantic models.
Ed taps Subject field dictates and corrects a couple of sentences explaining thc purpose of the meeting.	Bulk text can be entered easily and faster.

People tend to like to use speech to enter data and pen for corrections and pointing. As illustrated in Table 18.4, MiPad's Tap and Talk interface offers a number of benefits. MiPad has a Tap and Talk field that is always present on the screen, as illustrated in MiPad's start page in Figure 18.10 (a) (the bottom gray window is always on the screen).

The user can give spontaneous commands by tapping the Tap and Talk field and talking to it. The system recognizes and parses the command, such as showing a new appointment form. The appointment form shown on MiPad's display is similar to the underlying semantic objects. The user can have conversation by *tapping and talking* to any subfield as well. By tapping to the attendees field in the calendar card shown in Figure 18.10 (b), for example, the semantic information related to potential attendees is used to constrain both CSR and SLU, leading to a significantly reduced error rate and dramatically improved throughput. This is because the perplexity is much smaller for each subfield-dependent language and semantic model. General text fields, such as the title or body of an email, call for general dictation. Dictation is handled by the same mechanism as other speech entries: The user dictates while tapping the pen where the words should go. To correct a misrecognized word, a tap on the word invokes a correction window, which contains the n-best list and a soft keyboard. The user may hold the pen on the misrecognized word to dictate its replacement.

From the start page, you can reach most application states. There are also a number of short-cut states that can be reached via the hardware buttons. The design is refined incrementally in a number of iterations.

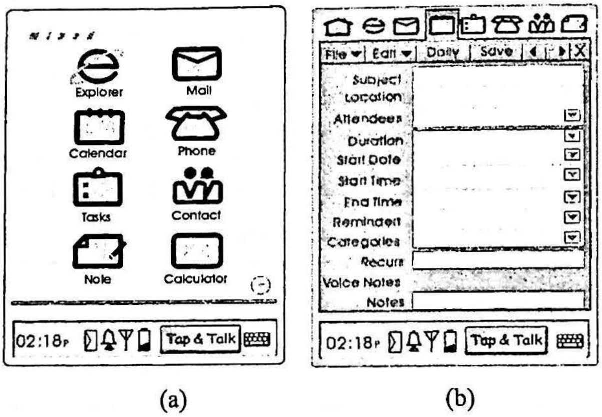


Figure 18.10 Concept design for (a) MiPad's main card and (b) MiPad's calendar card.

18.5.2. Rapid Prototyping

When you have the spec for the application, you need to communicate the design to the development team so they can rapidly develop a prototype for further iterations. In practice, handing off an interface to a development team without involvement in its implementation seldom works. This is because the developers always have questions about the design document. You need to be available to them to interpret your design. Technical and practical issues may also come up during development, requiring changes to the interface design.

For speech applications, there are a number of key components that have a significant implication for the user interface design. As your design iterations progress, it is useful to track the changes in these components and the effect that these changes have on the user experience of the prototype.

- **Semantic classes**: Each semantic class, as discussed in Chapter 17, is representing real-world entities or associated with the action that applications can take. We need to abstract the application to a level such that we can represent all the actions that the application can take with a number of semantic classes.

Each semantic class has needed slots that require a context-free grammar with a decent coverage. You can start with example sentences and reuse some of the predefined task-independent CFGs, such as date and time. The interface design can help to constrain users to say the sentences that are within the domain.

- **Dialog control**: When appropriate semantic objects are selected, you need to decide the flow of these semantic objects with an appropriate dialog strategy, either speech-based or GUI-based, which should include both inter- and intra-frame control and error-repair strategy. In addition, appropriate use of confidence measures is also critical.
- **Language models**: You need to use appropriate language models for the speech recognizer. Context-free grammars used for the semantic classes can be used for the recognizer. As coverage is generally limited, it is desirable to use the *n*-gram as an alternative. You can use a unified context-free grammar and *n*-gram for improved robustness, as introduced in Chapter 11.
- **Acoustic models**: In comparison to the semantic classes and language models, subphonetic acoustic models can be used relatively effectively for task-independent prototyping. Nevertheless, mismatches such as microphones, environment, vocabulary, and speakers are critical to the overall performance. You may want to consider speaker-dependent or environment-dependent prototyping initially, as this minimizes the impact of various mismatches.
- **TTS models**: If the speech output is well defined, you can adapt the general-purpose TTS systems with the correct prosody patterns. This can be achieved with the transplanted prosody in which variables can be reserved for TTS to deal only with proper nouns or phrases that cannot be predecided. You can also adapt acoustic units such that the concatenation errors can be minimized.

The work needed for rapid prototyping is significant, but less than it might at first appear to be. Earlier tasks can be repeated in later iterations when more data become available in order to further improve the usability of the overall system.

18.5.3. Evaluation

It is vital to have a rigorous evaluation and testing program to measure the usability of the prototype. It is a good practice to evaluate not only the entire system but also each individual component. The most important measure should be whether users are satisfied with the entire system. You can ask questions such as: *Is the task completion time much better? Is it easier to get the job done? Did we solve the major problem existing in the previous design?* Most of the time, we may find that the overall system can't meet your goal. This is why you need to further analyze your design and evaluate each individual component from UI to the underlying engines.

Before the prototype becomes fully functioning, you may want to use Wizard-of-Oz (WOZ) experimentation to study the impact of the prototype. That is, you have a real person (the wizard) to control some of the functions that you have not implemented yet. The person can carry out spoken interactions with users, who are made to believe that they are interacting with a real computer system. The goal of WOZ is to study the behavior of human-computer interactions. By producing data on the interaction between a simulated system and its users, WOZ provides the basis for early tests of the system and its feasibility prior to implementation. It can also simulate the error patterns in the existing spoken language system.

WOZ is still expensive, as you have the cost of training a wizard in addition to the usual costs of selecting experimental subjects, transcribing the session, and analyzing overall results. The use of WOZ has so far been justified in terms of relatively higher cost of having to revise an already implemented system that turned out to be flawed in the usability study.

In practice, WOZ may be replaced by the implementation-test-revise approach based on available tools. Whether or not WOZ is preferable depends on several factors, such as the methods and tools available, the complexity of the application to be designed, and the risk and cost of implementation failure. Low complexity speaks in favor of directly implementing the interface and application without interposing a simulation phase. On the other hand, high complexity may advocate iterative simulations before a full-fledged implementation.

To evaluate each component, you often need to select users that typify a wide range of potential users. You can collect representative data from these typical users based on the prototype we have. The most important measures include language-model perplexity, speech recognition accuracy and speed, parser accuracy and speed (on text and speech recognition output), dialog control performance, prompting strategy and quality, error-repair efficacy, impact of adaptive modeling, and interactions among all these components.

Effective evaluation of a user interface depends on a mix of objective and subjective measures. One of the most important characteristics is repeatability of your results. If you can repeat them, then any difference is likely due to the changes in the design that you are testing. You will want to establish reference tasks for your application and then measure the time that it takes subjects to complete each of those tasks. The consistent selection of subjects is important; as is the use of enough subjects to account for individual differences in the way they complete the tasks.

How do we evaluate MiPad? For our preliminary user study, we did a benchmark study to assess the performance in terms of task-completion time, text throughput (WPM), and user satisfaction. The focal question of this study is whether the MiPad interface can provide added value to the existing PDA interface.

Is the task-completion time much better? We had 20 computer-savvy users test the partially implemented MiPad prototype. These people had no experience with PDA or speech recognition software. The tasks we evaluated include creating a new e-mail, checking a calendar, and creating a new appointment. Task order was randomized. We alternated tasks to different user groups using either pen-only or Tap and Talk interfaces. The text throughput was calculated during email paragraph transcription tasks.[14] Compared to using the pen-only iPaq interface, we observed that the Tap and Talk interface is about twice as fast transcribing email.[15] For the overall command-and-control operations such as scheduling appointments, the Tap and Talk interface was about 33% faster than the existing pen-only interface.[16] Error correction for the Tap and Talk interface was one of the most unsatisfied features. In our user study, calendar access time using the Tap and Talk methods was about the same as pen-only methods, which suggests that simple actions are very suitable for pen-based interaction.

Is it easier to get the job done? Most users we tested stated that they preferred using the Tap and Talk interface. The preferences were consistent with the task completion times. Indeed, most users' comments concerning preference were based on ease of use and time to complete the task, as shown in Figure 18.11.

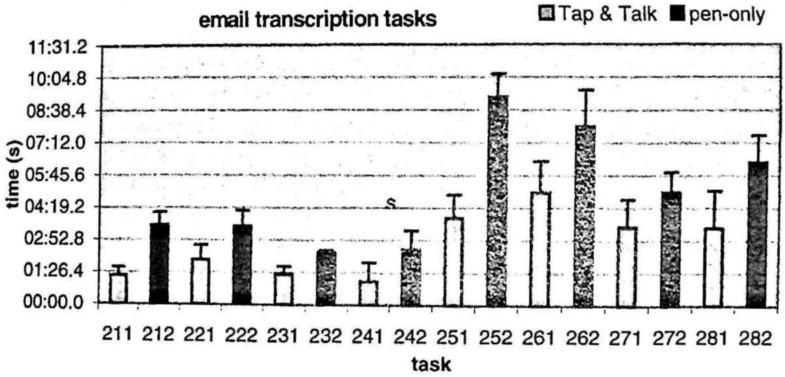


Figure 18.11 Task completion time of email transcription between the pen-only interface and Tap and Talk interface.

18.5.4. Iterations

With rigorous evaluations, you can revisit the components that are on the critical path. In practice, you need to run several iterations to improve user interface design, semantic classes, CFGs, *n*-gram, dialog control, and acoustic models. Iterative experiments are particularly important for deployment, as the coverage of the grammars can be dramatically improved only after we accumulate enough realistic data from end users.

[14] Transcription may not be a realistic task and gives an advantage to speech, because speech avoids a lot of attention shifting that is involved with pen input.

[15] The corresponding speech recognition error rate for the tasks is about 14%, which is based on using a close-talk microphone and a speaker-dependent acoustic model trained from about 20 minutes of speech.

[16] The SLU (at card level) error rate for the task is about 4%.

For MiPad, the performance improvement from using real data has been more dramatic than any other new technologies we can think of. When we collected domain-specific data, MiPad's SLU error rate was reduced by more than a factor of two. The major improvement came from broadening the semantic grammars from real data. We believe that data collection and iterative improvement should be an integral part of the process of developing interactive systems.

Despite unquestionable progress in the last decade, there are many unsolved problems in the procedures, concepts, theory, methods, and tools. Designing an effective speech application and interface remains as much of an art as it is an exact science with established procedures of good engineering practice. The quest for an effective speech application has been the goal for many speech and interface researchers. A successful speech application needs to significantly improve people's productivity. We believe that speech will be one of the most important technology components to enable people access information more effectively in the near future.

MiPad is a work in progress to develop a consistent interaction model and engine technologies for three broad classes of applications. A number of discussed features are yet to be fully implemented and iteratively tested. Our currently tested features include PIM functions only. Despite our incomplete implementation, we observed in our preliminary user study that speech and pen have the potential to significantly improve user experience. Thanks to the multimodal interaction, MiPad also offers a far more compelling user experience than standard telephony interaction. The success of MiPad depends on spoken language technology and always-on wireless connection. With upcoming 3G wireless deployments in sight,[17] the critical challenge for MiPad remains the accuracy and efficiency of our spoken language systems, since MiPad may be used in noisy environments without a close-talk microphone, and the server also needs to support a large number of MiPad clients.

18.6. HISTORICAL PERSPECTIVE AND FURTHER READING

User interface design is particularly resistant to rigid methodologies that attempt to define it simply as another branch of engineering. This is because design deals with human perception and human performance. Issues affecting the user interface design typically include a wide range of topics such as technology capability and limitations, product productivity (speed, ease of use, range of functions, flexibility), esthetics (look and feel, user familiarity, user impression), ergonomics (cognitive load, memory), and user education [2, 13, 14, 30, 35, 38, 44, 45]. The spoken language interface is a unique and specialized human interface medium that differs considerably from the traditional graphical user interface, and there are a number of excellent books discuss how to build an effective speech interface [1, 25, 39, 43].

When we communicate with speech, we must hear and comprehend all information sequentially. This very serial or sequential nature makes speech less effective as an information presentation medium than the graphics. As an input medium, speech must be supplied

[17] http://www.wirelessweek.com/issues/3G

one command or word at a time, requiring the user to remember and order information in a way that is more taxing than it is with visual and mechanical interfaces. The powerful and parallel presentation of information in today's GUI in the form of menus, icons, and windows makes GUI one of the most effective media for people to communicate. GUI is not only parallel but also persistent. That is, once presented, information remains on the screen until replaced as desired. Speech has no such luxury, which places the burden of remembering machine output onto the user. It is no wonder that *a picture is worth a thousand words*. We believe this is one of the key reasons why the spoken language interface has not taken off in comparison to the graphical user interface.

Although speech is natural, as humans already know how to talk effortlessly, this common shared experience of humans may not necessarily translate to human-machine interaction. This is because machines do not share a common cultural heritage with humans and they do not posses certain assumptions about the reality of the world in which we live. We have no precedent for effective speech interaction with nonsentient devices that are self-aware. Our experience with natural speech interaction is based on the assumption that when we talk to another person, the other person has some stake in the outcome of the interaction. The result of this expectation is that structured and goal-oriented conventions are necessary to steer people away from social speech behaviors toward task-oriented protocols, which essentially eliminate our expectation of naturalness.

The fact that speech interactions are one dimensional and time consuming has significant ramifications for interface design. The major challenge is to help the user accomplish as much as possible in as little time as possible. The designer thus must encourage the user to construct a mental model of the task and the interface that serves it, thereby creating an illusion of forward movement that exploits time by managing and responding to the user goals. Many telephony applications have carefully considered these constraints and provided a viable alternative to GUI-based applications and interfaces.

Despite the challenges, a number of excellent applications set milestones for the spoken language industry. Widely used examples include DECTalk (TTS), AT&T universal card services, Dragon's NaturallySpeaking dictation software, and Microsoft Windows 2000 TTS accessibility services. With the wireless smart devices such as smart phones, we are confident that MiPad-like devices will find their way to empower people to access information anywhere and anytime, leading to dramatically improved productivity.

There are a number of companies offering a variety of new speech products. Their Web sites are the best reference points for further reading. The following Web sites contain relevant spoken language applications and user interface technologies:

- AOL (http://www.aol.com)
- Apple (http://www.apple.com/macos/speech)
- AT&T (http://www.att.com/aspg)
- BBN (http://www.bbn.com)
- Dragon (http://www.dragonsystems.com)
- IBM (http://www.ibm.com/software/speech)

- Lernout & Hauspie (http://www.lhs.com)
- Lucent Bell Labs (http://www.lucent.com/speech)
- Microsoft (http://www.microsoft.com/speech)
- Nuance (http://www.nuance.com)
- NTT (http://www.ntt.com)
- Philips (http://www.speech.be.philips.com)
- Speechworks (http://www.speechworks.com)
- Tellme (http://www.tellme.com)
- Wildfire (http://www.wildfire.com)

REFERENCES

[1] Balentine, B. and D. Morgan, *How to Build a Speech Recognition Application*, 1999, Enterprise Integration Group.

[2] Beyer, H. and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*, 1998, San Francisco, Morgan Kaufmann Publishers.

[3] Campbell, J., "Speaker Recognition: A Tutorial," *Proc. of the IEEE*, 1997, **85**(9), pp. 1437-1462.

[4] Chang, E., *et al.*, "Large Vocabulary Mandarin Speech Recognition with Different Approaches in Modeling Tones," *Int. Conf. on Spoken Language Processing*, 2000, Beijing, China.

[5] Chien, L.F., K.J. Chen, and L.S. Lee, "A Best-First Language Processing Model Integrating the Unification Grammar and Markov Language Model for Speech Recognition Applications," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(2), pp. 221-240.

[6] Dobler, S., "Speech Control In The Mobile Communications Environment," *IEEE ASRU Workshop*, 1999, Keystone, CO.

[7] Dugast, C., X. Aubert, and R. Kneser, "The Philips Large-Vocabulary Recognition System for American English, French and German," *Proc. of the European Conf. on Speech Communication and Technology*, 1995, Madrid, pp. 197-200.

[8] Eichner, M. and M. Wolff, "Data-Driven Generation of Pronunciation Dictionaries In The German Verbmobil Project - Discussion of Experimental Results," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, pp. 1687-1690.

[9] Furui, S., "Recent Advances in Speaker Recognition," *Pattern Recognition Letters*, 1997, **18**, pp. 859-872.

[10] Galitz, W.O., *Handbook of Screen Format Design*, 1985, Wellesley, MA, Q. E. D. Information Sciences Inc.

[11] Galitz, W.O., *User-Interface Screen Design*, 1993, Wellesley, MA, Q. E. D. Information Sciences Inc.

[12] Gao, J., *et al.*, "A Unified Approach to Statistical Language Modeling for Chinese," *Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, pp. 1703-1706.

[13] Hackos, J. and J. Redish, *User and Task Anaysis for User Interface Design*, 1998, New York, John Wiley.

[14] Helander, M., *Handbook of Human-Computer Interaction*, 1997, Amsterdam, North-Holland.

[15] Hon, H.W., *et al.*, "Towards Large Vocabulary Mandarin Chinese Speech Recognition," *Int. Conf. on Acoustics, Signal and Speech Processing*, 1994, Adelaide, Australia, pp. 545-548.

[16] Hon, H.-W., Y. Ju, and K. Otani, "Japanese Large-Vocabulary Continuous Speech Recognition System Based on Microsoft Whisper," *The 5th Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[17] Horton, W., *The Icon Book*, 1994, New York, John Wiley.

[18] Huang, X., *et al.*, "MIPAD: A Next Generation PDA Prototype," *Int. Conf. on Spoken Language Processing*, 2000, Beijing, China.

[19] Karat, C., *et al.*, "Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems," *CHI'99*, 1999, Pittsburgh, pp. 15-20.

[20] Karat, J., "Int. Perspectives: Some Thoughts on Differences between North American and European HCI," *ACM SIGCHI Bulletin*, 1991, **23**(4), pp. 9-10.

[21] Kumamoto, T. and A. Ito, "Structural Analysis of Spoken Japanese Language and its Application to Communicative Intention Recognition," *Proc. of the Fifth Int. Conf. on Human-Computer Interaction—Poster Sessions: Abridged Proc.*, 1993, pp. 284.

[22] Lamel, L. and R.D. Mori, "Speech Recognition of European Languages" in *Proc. IEEE Automatic Speech Recognition Workshop* 1995, Snowbird, UT, pp. 51-54.

[23] Laurila, K. and P. Haavisto, "Name Dialing—How Useful Is It?," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2000, Istanbul, pp. 3731-3734.

[24] Lyu, R.Y., *et al.*, "Golden Mandarin (III)—A User-Adaptive Prosodic Segment-Based Mandarin Dictation Machine for Chinese Language with Very Large Vocabulary," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995, Detroit, pp. 57-60.

[25] Markowitz, J., *Using Speech Recognition*, 1996, Upper Saddle River, Prentice Hall.

[26] Martin, G.L., "The Utility of Speech Input in User-Computer Interfaces," *Int. Journal of Man-Machine Studies*, 1989, **30**(4), pp. 355-375.

[27] Möbius, B., "Analysis and Synthesis of German F0 Contours by Means of Fujisaki's Model," *Speech Communication*, 1993, **13**(53-61).

[28] Morrison, D.L., *et al.*, "Speech-Controlled Text-Editing: Effects of Input Modality and of Command Structure," *Int. Journal of Man-Machine Studies*, 1984, **21**(1), pp. 49-63.

[29] Nielsen, J., *Usability Engineering*, 1993, Boston, MA, Academic Press.

[30] Norman, D.A., *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*, 1993, Reading, MA, Addison-Wesley.

[31] O'Shaughnessy, D., "Specifying Accent Marks in French Text for Teletext and Speech Synthesis," *Int. Journal of Man-Machine Studies*, 1989, **31**(4), pp. 405-414.

[32] Pierrehumbert, J., and M. Beckman, *Japanese Tone Structure*, 1988, Cambridge, MA, MIT Press.

[33] Poock, G.K., "Voice Recognition Boosts Command Terminal Throughput," *Speech Technology*, 1982, **1**, pp. 36-39.

[34] Rao, P.V.S., "VOICE: An Integrated Speech Recognition Synthesis System for the Hindi Language," *Speech Communication*, 1993, **13**, pp. 197-205.

[35] Royer, T., "Using Scenario-Based Designs to Review User Interface Changes and Enhancements," *Proc. of DIS'95 Symposium on Designing Interactive Systems: Processes, Practices, Methods, & Techniques*, 1995, pp. 237-246.

[36] Rubin, J., *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, 1994, New York, John Wiley.

[37] Sagisaka, Y. and L.S. Lee, "Speech Recognition of Asian Languages" in *Proc. IEEE Automatic Speech Recognition Workshop* 1995, Snowbird, UT, pp. 55-57.

[38] Salvendy, G., *Handbook of Human Factors and Ergonomics*, 1997, New York, John Wiley.

[39] Schmandt, C., *Voice Communication with Computers*, 1994, New York, Van Nostrand Reinhold.

[40] Schmidt, M., *et al.*, "Phonetic Transcription Standards for European Names (ONOMASTICA)," *Proc. of the European Conf. on Speech Communication and Technology*, 1993, Berlin, pp. 279-283.

[41] Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 1997, Reading, MA, Addison-Wesley.

[42] Spencer, R.H., *Computer Usability Testing and Evaluation*, 1985, Englewood Cliffs, NJ, Prentice-Hall.

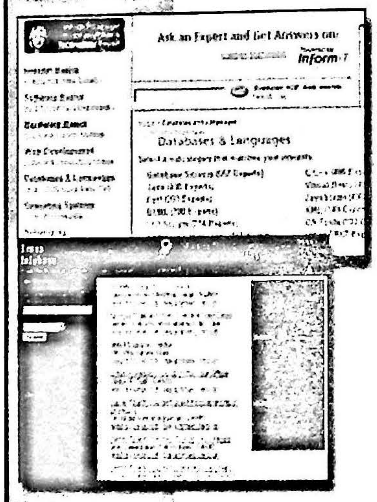
[43] Weinschenk, S. and D. Barker, *Designing Effective Speech Interfaces*, 2000, New York, John Wiley.

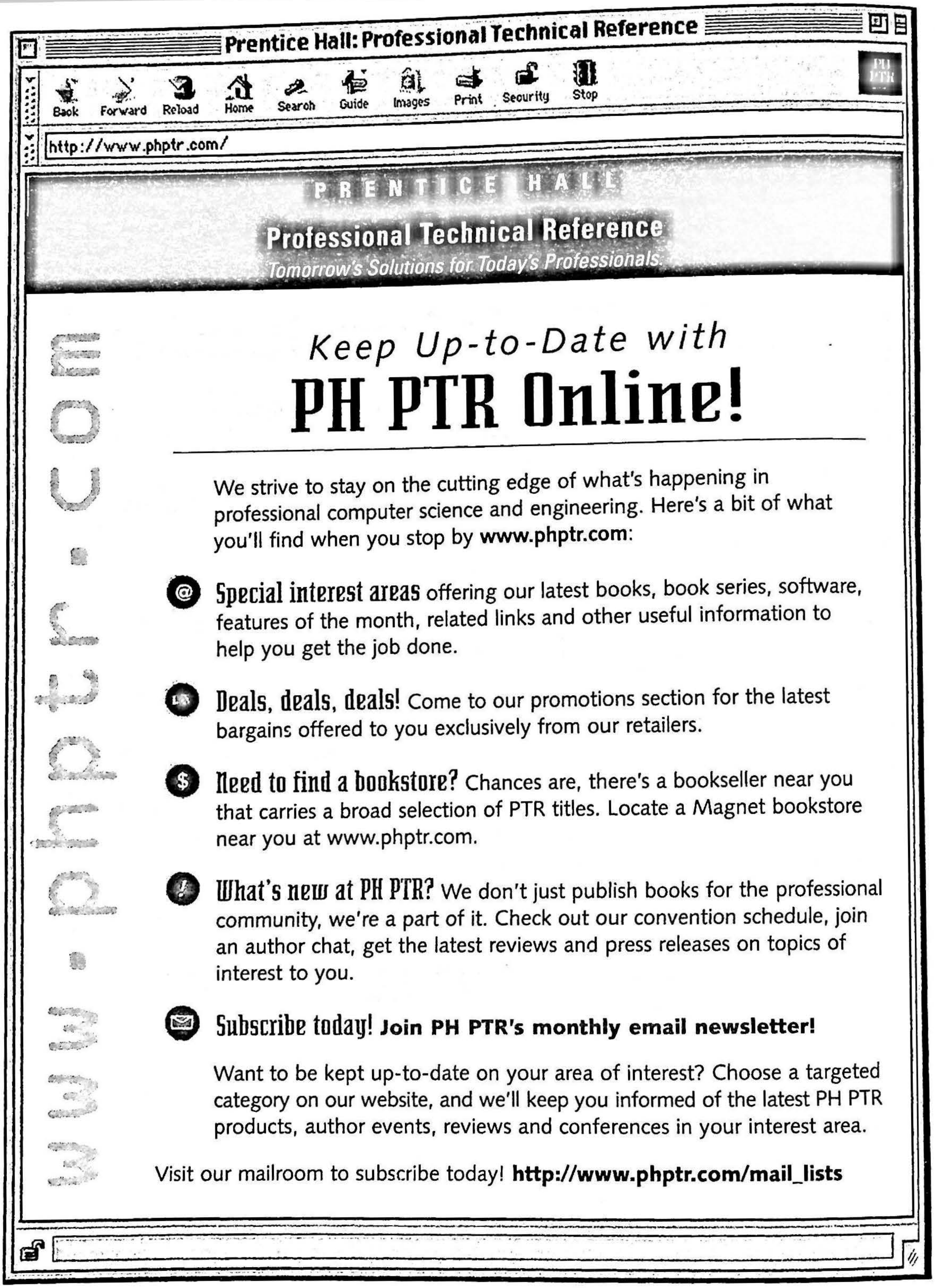
[44] Wixon, D. and J. Ramey, *Field Methods Casebook for Software Design*, 1996, New York, John Wiley.

[45] Wood, L., *User Interface Design: Bridging the Gap from User Requirements to Design*, 1997, CRC Press.

[46] Yang, K.C., *et al.*, "Statistics-Based Segment Pattern Lexicon—A New Direction for Chinese Language Modeling," *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, pp. 169-172.

INDEX


Ask an Expert and Get Answers on:
InformIT
Databases & Languages


Prentice Hall: Professional Technical Reference
Back
Forward
Reload
Home
Search
Guide
Images
Print
Security
Stop
PH PTR
http://www.phptr.com/
PRENTICE HALL
Professional Technical Reference
Tomorrow's Solutions for Today's Professionals.
www.phptr.com

Keep Up-to-Date with

PH PTR Online!

We strive to stay on the cutting edge of what's happening in professional computer science and engineering. Here's a bit of what you'll find when you stop by **www.phptr.com**:

- **Special interest areas** offering our latest books, book series, software, features of the month, related links and other useful information to help you get the job done.
- **Deals, deals, deals!** Come to our promotions section for the latest bargains offered to you exclusively from our retailers.
- **Need to find a bookstore?** Chances are, there's a bookseller near you that carries a broad selection of PTR titles. Locate a Magnet bookstore near you at www.phptr.com.
- **What's new at PH PTR?** We don't just publish books for the professional community, we're a part of it. Check out our convention schedule, join an author chat, get the latest reviews and press releases on topics of interest to you.
- **Subscribe today!** **Join PH PTR's monthly email newsletter!**

Want to be kept up-to-date on your area of interest? Choose a targeted category on our website, and we'll keep you informed of the latest PH PTR products, author events, reviews and conferences in your interest area.

Visit our mailroom to subscribe today! **http://www.phptr.com/mail_lists**

Voice/Speech
Recognition Technology

XUEDONG HUANG

ALEX ACERO

HSIAO-WUEN HON

SPOKEN LANGUAGE PROCESSING

A Guide to Theory, Algorithm, and System Development

- New advances in spoken language processing: *theory and practice*
- In-depth coverage of speech processing, speech recognition, speech synthesis, spoken language understanding, and speech interface design
- Many case studies from state-of-the-art systems, including examples from Microsoft®'s advanced research labs

Spoken Language Processing draws on the latest advances and techniques from multiple fields: computer science, electrical engineering, acoustics, linguistics, mathematics, psychology, and beyond. Starting with the fundamentals, it presents all this and more:

- Essential background on speech production and perception, probability and information theory, and pattern recognition
- Extracting information from the speech signal: useful representations and practical compression solutions
- Modern speech recognition techniques: hidden Markov models, acoustic and language modeling, improving resistance to environmental noises, search algorithms, and large vocabulary speech recognition
- Text-to-speech: analyzing documents, pitch and duration controls, trainable synthesis, and more
- Spoken language understanding: dialog management, spoken language applications, and multimodal interfaces

To illustrate the book's methods, the authors present detailed case studies based on state-of-the-art systems, including Microsoft's Whisper speech recognizer, Whistler text-to-speech system, Dr. Who dialog system, and the MiPad handheld device. Whether you're planning, designing, building, or purchasing spoken language technology, this is the state of the art—from algorithms through business productivity.

About the Authors

XUEDONG HUANG is founder and head of the Speech Technology Group at Microsoft Research. He received his Ph.D. from the University of Edinburgh. He is an IEEE Fellow.

ALEX ACERO and HSIAO-WUEN HON are Senior Researchers at Microsoft Research and Senior Members of IEEE. Both received doctorates from Carnegie Mellon University.

PRENTICE HALL
Upper Saddle River, NJ 07458
www.phptr.com

ISBN 0-13-022616-5
90000
9 780130 226167
V036 51010045138