

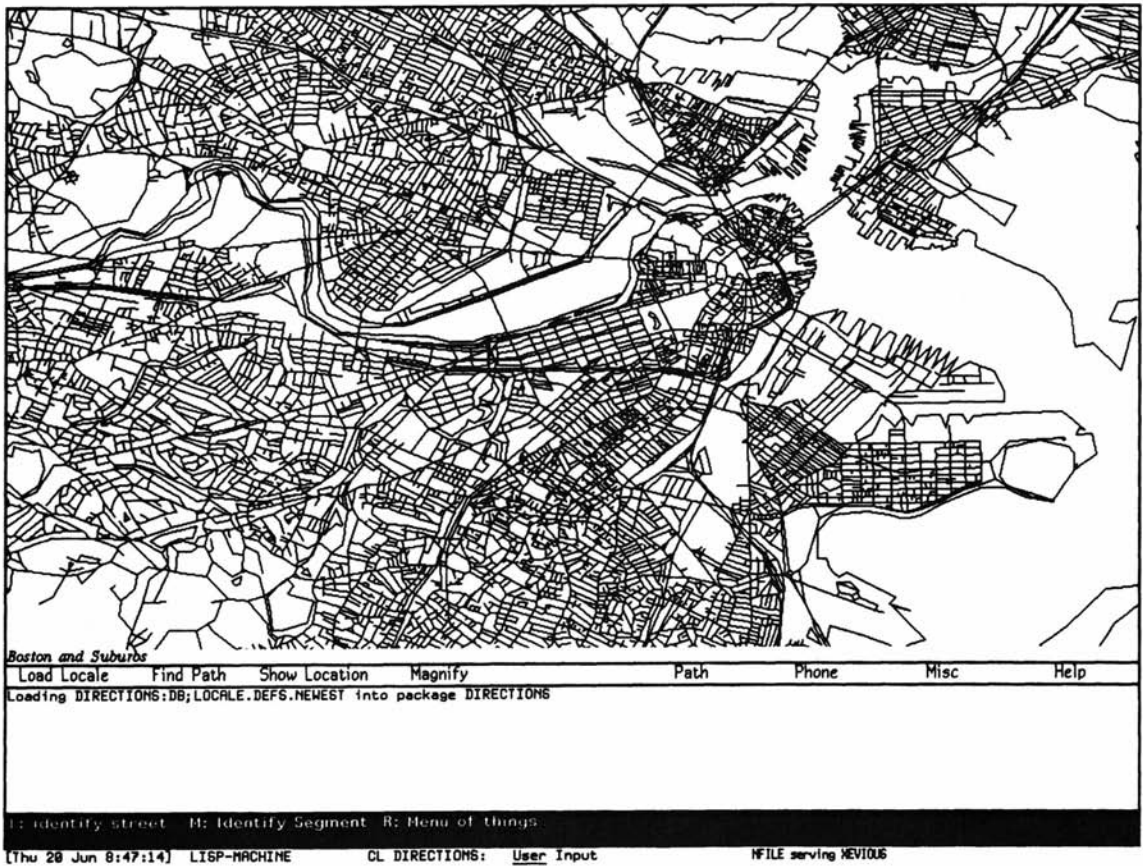


Figure 6.5. The streets in Direction Assistance's database.

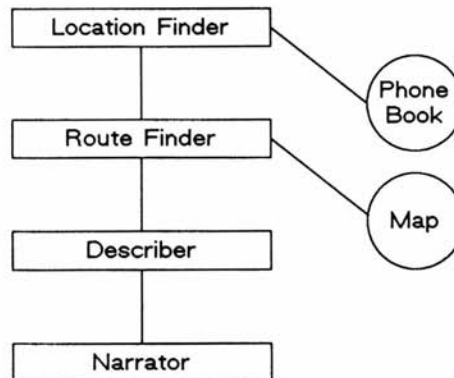


Figure 6.6. Major software modules in Direction Assistance.

for both house number and street name and informs the user to use the “#” key after entering the house number. Of course, explicit termination can fail if the user neglects to use the termination key. If the user pauses, waiting for a response, the program indicates that it thinks the caller is still entering an address and notes that a terminator key is required. If the user forgets the terminator and simply continues, next entering the street name, the system may detect this. When the number of digits entered exceeds the highest known house number in the database, the system indicates that this is an unknown address, reminds the caller to terminate address entry with a pound sign, and starts over.

Entering a street name is more challenging due to multiple instances of the same street name in the database. The caller is prompted to spell the name of the street using one telephone key per character. Again explicit termination is requested. In the case of multiple streets having identical touch tone “spellings,” Direction Assistance resorts to a temporal menu of the possible choices. However, much more common than spelling confusion is that the same street may be found in multiple municipalities, or the same name can apply to multiple streets even in one municipality (e.g., Hancock Street, Hancock Place, and Hancock Park in Cambridge). In these circumstances, Direction Assistance again resorts to a temporal menu to finish the selection. To help explain the selection process, the system first indicates why it is confused, e.g., “I don’t know whether you mean Hancock Street, Place, or Park. Hit any key when you hear the one you want.”

Route Description

Once the origin and destination are known, the Route Finder determines a route. The route consists of a path traversing a number of street segments (each segment being approximately one block long). The route is optimized on the basis of distance, quality of the streets, ease of driving (e.g., a penalty is imposed for left turns), and ease of explanation (a zig-zag route may be shorter but may be more complicated and difficult to follow).

The next step is the most challenging language-generation task of this application. The Describer must take the series of segments from the database and turn it into a coherent description of how to drive to the destination. A list of what to do at each intersection is hardly useful to the driver and interferes with the goal of speaking succinctly. Instead the Describer generates instructions comparable to those a person gives. A passenger assumes the driver takes the “obvious” route (e.g., stays on the main street) until told otherwise. This allows the Describer to compress many sequential segments into a single driving act, e.g., “Stay on Main Street until it merges with Massachusetts Avenue.”

The Describer employs a taxonomy of intersection types including such classes as **continue**, **forced-turn**, **turn**, **fork**, **onto-rotary**, and **exit-rotary**. A software module associated with each is consulted in order at each intersection. The appropriate module chooses to describe the driving act to perform at this intersection. Succinctness is achieved by having the **continue** expert usually choose to say nothing unless a street name changes or a significant landmark is encountered.

The output of the Describer is the text description of a “tour,” an example of which follows.

“If your car is parked on the same side of the street as 20 Ames Street, turn around and start driving. Drive all the way to the end and take a left onto Memorial Drive. After about one quarter of a mile, take the easy left and merge onto Main Street. It becomes the Longfellow Bridge. After you cross the bridge you will come to a rotary. Go about three quarters of the way around it, and turn onto Charles Street. Merge with Storow Drive.”

The text of the tour must be recited by a speech synthesizer; this is done by the Narrator. It is the Narrator’s responsibility to recite the tour at a rate that allows the user to write down the directions. Unfortunately people write directions at greatly varying rates; some have elaborate shorthand notations and can record the directions quickly, while others write them word-by-word. The Narrator’s only recourse is to pause between each part of the route and wait for touch tone input from the user as a flow-control mechanism, but this does not make for a very comfortable interface. More graceful flow control requires spoken feedback from the direction recipient; an attempt at this is described in Chapter 9.

Back Seat Driver

Back Seat Driver is a subsequent Media Lab project that also employs speech synthesis for direction giving but in the very different context of operating in a car in real time while the user is driving. Although much of the software from Direction Assistance proved useful to this project, Back Seat Driver is different in two respects. First is the motivation to use voice because a driver’s eyes are busy watching the road, whereas Direction Assistance focused on remote telephone access. This raises the issue of how to instill user confidence in a system that cannot be seen and speaks only when necessary. Second is the real-time nature of Back Seat Driver; it must know when to speak, how long it will take to say an utterance, and detect when the driver has made a mistake.

Back Seat Driver utilizes a vehicle navigation system to report the automobile’s position, speed, and, direction to the route planner.⁹ This navigation system uses an inertial guidance system in combination with the constraints provided by the map database (which is stored on CD-ROM) to determine position; for more discussion of the various location determining technologies, see [Davis 1989]. Route planning and discourse generation are done by software running on a computer in the trunk speaking via a speech synthesizer connected to one of the serial ports of the computer (see Figure 6.7).

The initial task of the Back Seat Driver is identical to that of Direction Assistance, namely, to elicit the destination from the driver (the current position of the

⁹The vehicle navigation system was built by the project’s sponsor, Nippon Electric Company.

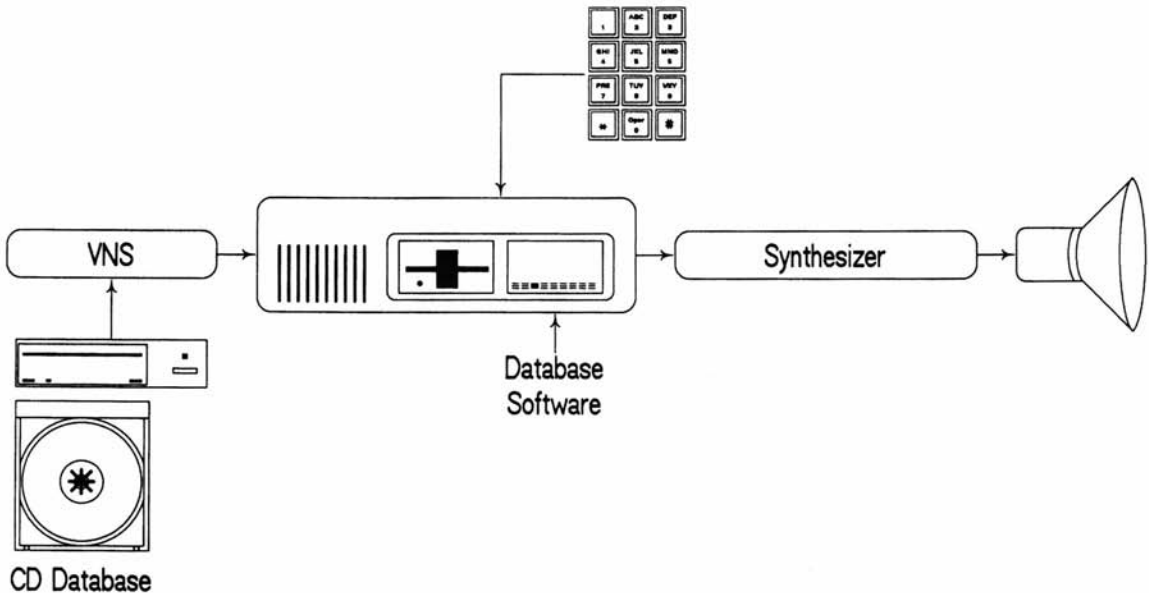


Figure 6.7. Hardware configuration of the Back Seat Driver.

car is known to the system) and plan a route. In various implementations, the destination has been entered by means of a telephone keypad (an interface identical to that used by Direction Assistance) or a computer keyboard. Having obtained the destination, Back Seat Driver plans a route, speaks the initial instruction (e.g., “Turn around and start driving”), and waits for the driver to proceed. From this point on, its task is to provide spoken directions, as they are needed, for each segment of the trip.

The goal of Back Seat Driver is to get the user to the destination; this is facilitated by the secondary task of keeping the driver confident that the system is working and correct. These tasks are closely related, both requiring a richer discourse capability than Direction Assistance as well as an improved method of time management.

Getting the user to a destination requires careful phrasing of the directions. A major problem with the use of synthetic speech for this task is that some street names may not be pronounced correctly. Names are also less useful than other cues in Boston due to the lack of proper street labeling. In an early version, Back Seat Driver would say “Take a left at Fulkerson Street”; this was changed to “Take the next left; it’s Fulkerson Street.” The driving action is described in a simple sentence for ease of intelligibility [Davis and Schmandt 1989]. The street name becomes a secondary cue, and the drive can still follow the directions even if the name is badly mispronounced or the street sign is missing.

Back Seat Driver also offers advice and warnings to make the drive safer and increase the probability of the driver completing the requested turn. For example, it may say “Get into the left-hand lane because you are going to take a left at the

next set of lights” or warn when a one-way street becomes two way. To provide adequate warning, each step of the directions is repeated twice: once about 30 seconds before the intersection in question and another just before the decision point.

Precise timing of detailed utterances such as “Take a left at these lights” just as the vehicle approaches the intersection, along with the use of deictic pronouns¹⁰ help instill confidence in the driver. Because there is no display and Back Seat Driver does not speak more often than is necessary, it is easy to doubt that it is actually “alive.” Utterances of this type affirm that the system is tracking the vehicle’s location correctly and will speak when necessary.

To speak at the right time Back Seat Driver needs to manage time in concert with the real world of a vehicle in motion. Time management is achieved by maintaining a series of goals with an estimated time to speak each goal. The time at which an utterance must commence changes relative to the vehicle’s speed, and there may not be adequate time to speak all goals. The goals are prioritized so that the most important are spoken first. A goal that would be interrupted by a higher-priority goal (given an anticipated moment to speak the goal) will instead choose not to start as it would be confusing for the system to interrupt itself as well as waste valuable time. Knowing how long it will take to speak a goal enables advance planning and allows some goals to be merged if there is not enough time for each in isolation [Davis and Schmandt 1990]. For example, just before a short jog in the road, Back Seat Driver might say “Take the left at the end of the road, and then you’re going to take the very next right.”

Goals are dynamic and can change during the course of the trip because the driver does not always follow the directions correctly. Back Seat Driver must cope with these driver errors. It indicates the mistake (“Oops, I meant for you to take a right . . .”) but then adapts to the error by computing a new route after telling the driver (“. . . but that’s OK, I’ll figure out a new way to get there”). The Route Finder can operate incrementally while the vehicle is in motion although the driver is told “Slow down while I think about it.” All the utterances were carefully chosen during a long iterative design cycle; it is inappropriate to speak in a manner that encourages the driver to simply slam on the brakes possibly causing an accident.

The primary focus of Back Seat Driver has been to experiment with discourse generation in real time so as to determine the appropriateness of speech output for a computer driving assistant. The project was based on the belief that voice could be employed effectively in a real-time application by implementing a goal-oriented discourse model. Although it was never rigorously tested against a display-based system, many users who test-drove the car found it effective and even reassuring. This was achieved in large part by the careful wording and timing of each utterance to match the car’s position and speed.

¹⁰Deictics are pronouns that refer to objects identified by their position relative to the speaker. “This” and “these” refer to nearby objects, “that” and “those” to distant objects.

Voiced Mail

Most telephone-based systems employing speech output involve a process of explicit user selection followed by a system response often repeated step-by-step while the user performs several operations. These systems spend a substantial portion of their time reciting menus. The menus may be hierarchical requiring the user to traverse several submenus and enter multiple key presses to achieve the desired action. While it is likely that a hierarchical structure may be well suited for the novice or occasional user, the experienced user learns the sequences and, provided the menus are interruptible, never hears much of them; callers are still often annoyed to key-press their way through the thick layering of commands to get to the information being sought.

This section presents as a case study a system that took a rather different view of user interaction. Voiced Mail [Schmandt 1984] was a program developed at M.I.T. to allow telephone access to electronic mail using speech synthesis. Voiced Mail provided a streamlined user interface oriented towards experienced users. It tried to minimize the time and attention required to navigate the user interface by avoiding automatic recitation of menus. Instead, the application followed a default sequence of presenting information about each message, then the message itself. The small number of commands were always active, and the application was always interruptible.

Voiced Mail applied extensive filtering to messages to remove extraneous information, made presentation decisions based on the length of each message, and rearranged message order to simplify presentation. These tasks were performed to minimize the time and attention required to use the application. Voiced Mail also employed a repetition strategy to help cope with the vagaries of synthesizing free-form, human-authored text.

Voiced Mail illustrates some advantages of incorporating multiple speech technologies into a single application. In Voiced Mail's later stages, recorded voice was used for generating replies to text messages as the keypad was inadequate to select between a limited repertoire of precomposed text messages. The synergy from employing multiple channels for computer interaction and supporting multimedia databases is a theme to which we will return in Chapter 12.

Flow of Control

To access Voiced Mail, a user dialed in and logged onto the system using his or her computer login ID and a password. The application would scan the user's new mail messages and decide how to present them. A default path was then taken through the messages; with no additional user input, the system would read each message in order to the caller. At no point was the user confronted with an explicit "If you want me to do . . . (some action), press a key"; rather, an active presentation was started, which could be interrupted at any time. There were places where the system would pause a moment to encourage possible user input, but if none was detected it would continue reading messages. This was all part of the strategy of avoiding menus.

Instead of using an explicit menu, Voiced Mail used an implicit set of commands that were always active and interruptible. These commands allowed forward or backward movement among messages or senders and invoked a few other functions. Since there were no explicit menus, each digit on the keypad was assigned to one command (the keypad mapping is shown in Figure 6.8.) This was a principled and deliberate choice because ease of use was more important than the number of functions supported by the application.

Voiced Mail sorted messages according to the sender and then presented all messages from the same sender sequentially, beginning with the sender who had sent the most messages.¹¹ Most mail systems present messages in the order in which they are received, but time-sequential messages bear little relationship to each other. Sequential messages from the same sender are often related; this ordering tried to preserve such context. A more thorough strategy would have been to identify a “thread” of messages with the same subject line (generated automatically by the “reply” command found with most text-based electronic mail systems) but from various senders.

After Voiced Mail determined presentation order, messages would be played one by one. If the message was brief, it was played immediately. If it was long (as measured by number of words), the system would warn “It’s a long message, about . . .” and recite the subject line. The system would then pause to invite user input (for example, to skip the long message). In the absence of any input, the message body would be played.

¹¹It would have been useful to have included an optional user profile that identified important message senders by name and played their messages first. This is an aspect of the filtering approach mentioned later and was implemented in a later email reading program.

1 Next Message	2 Previous Message	3 Repeat
4 Next Sender	5 Previous Sender	6 More Info
7 Yes	8 No	9 Reply
* Cancel	0 Pause/ Continue	# Quit

Figure 6.8. The command set for Voiced Mail.

This design was based on the observation that it would take more time to ask the user “Do you want to hear this message?” and wait for a response than to just play the message. The subject was not recited for the very short messages, as it was often redundant to the message body. This, however, was a mistake for some messages, e.g., that shown in Figure 6.9. A sample user interaction is shown in Figure 6.10.

Text Presentation Strategies

Because it avoided menus Voiced Mail spent most of its time actually reading messages. Recitation of a message required some careful filtering and an effective presentation strategy. There is much useless header information in a mail message. There may be punctuation or spacing conventions that should be preserved if possible, e.g., a double space between paragraphs is “spoken” as a short pause. Since there may also be typographic errors or words in the message that the synthesizer mispronounces or the user may need to hear a sentence again, a single linear pass through the message may be insufficient: the presentation must be interactive.

Figure 6.11 shows the full text of an electronic mail message including the rather voluminous header. Many fields, such as the message ID and the chain of hosts through which the message passed during transmission, are pointless to synthesize so they must be removed. Voiced Mail went further and decided that the only fields in the header that were vital were the sender and possibly the subject. A “more info” key offered more information, such as the full name of the sender and when the message was received. The time data was processed to produce text such as “Yesterday morning at 10” instead of “Wednesday 25 August 1993 9:57:32 EDT,” which both takes longer to speak and is more difficult to follow.

For the sake of its repetition strategy (see below), Voiced Mail parsed the message into sentences. To aid clarity, a pause was introduced between each sentence; this pause was about twice as long as would ordinarily be produced by the synthesizer. A pause twice as long again was used at each paragraph in the message body as the paragraph structure is also important for understanding the message’s content.

```
Message 141:
From derek Thu Mar 15 18:05:25 1990
To: geek
Subject: Metal-Oxide Varistor
Date: Thu, 15 Mar 90 18:05:21 EST
```

```
It's here . . . I'll leave it in the sound room on the middle shelf . . .
```

Figure 6.9. A mail message for which the subject is required to understand the message.

"Welcome to Voiced Mail. Please log in."

user presses 4-3-3-5 (sequence for "geek", the user's login).

"Hello Chris Schmandt. You have twelve new messages. <pause>

Four from Walter Bender. <pause>

Message one. This is a very long message; it's about window system subroutines . . ."

user presses 1 (next message).

"Message two. It's about the new conference room. <pause>

Can we get together tomorrow afternoon to decide about chairs?"

user presses 7 (affirmative reply).

"Message sent. Message three. It's about Monday's demo . . ."

user presses 4 (next sender).

"Two messages from Ben Stoltz. <pause>

Message one. It's about the driver bug . . ."

Figure 6.10. A sample Voiced Mail session.

Voiced Mail compromised intelligibility for speed by playing text somewhat faster than the default synthesis rate specified by the synthesizer hardware. At any moment, the user could invoke the **repeat** command. This command would cause the sentence currently being spoken (or which had just been spoken) to be repeated at 75% speed. If **repeat** was invoked again on the sentence, the sentence was spelled letter by letter.

Spell mode is so slow as to be nearly useless, however, and it is also quite difficult to follow. An alternate strategy, developed shortly after Voiced Mail, passes the message body through the local spelling checker dictionary. Those words not found in the dictionary are spelled. This will catch most typos as well as many of the proper nouns that might be particularly challenging to synthesize. For example, the phrase, "Schmandt debugged the software yesterday morning" would be pronounced "S-c-h-m-a-n-d-t debugged the s-o-f-t-w-a-r-e yesterday morning."

Advantages of Multiple Media

One of the advantages of electronic mail is the ease of posting a reply to the sender of a message. While reading mail, one is usually positioned at a screen and keyboard, which also provide the user interface for typing a reply. A single

From dcj@Sun.COM Tue Sept 22 15:53:21 1991
Received: by media-lab (5.57/4.8) id AA11928; Tue, 22 Sept 91
15:53:15 EDT
Received: from snail.Sun.COM (snail.Corp.Sun.COM) by Sun.COM (4.1/SMI-
4.1)
id AA24516; Tue, 22 Sept 91 12:53:51 PDT
Received: from jacksun.sun.com by snail.Sun.COM (4.1/SMI-4.1)
id AB18658; Tue, 22 Sept 91 12:51:56 PDT
Message-Id: <9109221953.AA06751@jacksun.sun.com>
To: Chris Schmandt <geek@media-lab.media.mit.edu>
Subject: possible bug in fax.c
In-Reply-To: Your message of Tue, 15 Sept 91 12:43:41 -0400.
<9109151643.AA27528@media-lab>
Date: Tue, 22 Sept 91 12:53:13 PDT
From: dcj@Sun.COM
Status: RO

I received your sound attachment email just fine.
This is a nice addition to Phoneshell!

Figure 6.11. An electronic mail message including its full header.

keystroke to the mail reader (“R” for “Reply”) initiates a reply with the return address generated automatically. Since the Email medium is asynchronous, the sender need not worry whether the recipient is awake to receive the message in whatever time zone the destination lies.

It was important for Voiced Mail to support a reply mechanism as well, so three of the twelve telephone keys were devoted to reply commands. One key would generate a message of the form “I read your message about . . . My answer is yes” and another generated a similar negative reply. A third key prompted the user for a phone number and sent a “please call me at . . .” message. Unfortunately, replies are rarely so unconditional as a simple yes or no. Also, when traveling one’s location and hence phone number are often transient, so the reply mechanism was enhanced to include the ability to record a voice message. The party who sent the original message would receive a computer-generated text mail message containing instructions on how to access the reply, which would involve calling the Voiced Mail telephone number and entering a special code at the login prompt. Upon hearing the message, the caller could then leave another voice recording in reply.

Of course, the recipient of the recorded message might be another subscriber to Voiced Mail. In this case, while reading mail from a terminal the recipient would see the same text message as delivered to the non-subscriber, i.e., he or she would have to call in to hear the voice message. If the subscriber were reading mail with Voiced Mail, the text message would never be seen as the system would simply play the recording. Such a unified interface to both voice and text messages was extended and incorporated into the Phone Slave conversational answering machine described in Chapter 11.

An important advantage of merging voice and text messages is the elimination of the need to consult multiple devices (telephone for voice mail, computer terminal for text mail) to find out whether any new messages have arrived. In the case of Voiced Mail, the initial motivation for including voice messages was the new reply capability required by telephonic access; text messages could be turned to voice, but voice replies could not be converted to text. This, therefore, is a case in which a new user interface medium results in the need to support multiple media in the underlying database being accessed.

Voiced Mail Today

Voiced Mail addressed universal access to electronic mail messages. At the time it was built (1983) modems were still 300 baud and computer terminals at home uncommon even at a research institution like M.I.T. But even under these circumstances, synthesis was too tedious to use for longer messages. Instead, users would scan messages to see if they had anything important to deal with (such as scheduling lunch) before coming into work. The exception was when users were traveling and had no other access to their mail.

Modem speeds have increased and terminals or personal computers at home are now commonplace; this decreased the need to use speech synthesis to access mail messages at home (although we do like to joke about hearing mail while in the shower). An interface such as Voiced Mail is more valuable to mobile users who may want to call in from airport lobbies or roadside phones. Voice is also particularly useful when used while doing other activities. One example is using a cellular telephone to scan mail while driving to work (the morning commute otherwise being wasted time.) A safer version of this scenario would involve uploading email as text into a laptop computer over telephone lines in the morning and then synthesizing it in the car while using speech recognition (instead of touch tones) to navigate among messages.

Speech synthesis access to Email over the telephone remains in surprisingly high demand at the Media Lab, and it has been included in a more integrated messaging environment. Phoneshell, which as described in Chapter 12, brings increased functionality to Email access including the ability to record a voice reply, to type a text reply using the telephone keypad, to send copies of replies to multiple parties chosen from an address listing, and to send a message to a third party as a fax. Another feature of the newer version is the ability to *filter* mes-

sages into broad categories (e.g., "urgent," "important," "personal," or "junk") based on pattern matching against email header fields (such as sender and subject) and the message body. Email filtering was first introduced by Malone in the Information Lens and has been shown to be an effective means of managing mail [Malone *et al.* 1987, Mackay *et al.* 1989]. For Phoneshell users, filtering makes it possible to attend to the most important messages first, which is essential when dealing with a slow communication channel.

Although the ability to reply and sort messages is crucial for the current success of speech access to email, the primary reason Email is popular is simply the ease of dialing in. Although most of Phoneshell's present users possess computers and modems at home, telephone access allows casual and low-overhead message retrieval from locations such as stores, pay phones in airports, other offices on campus, or even the kitchen when one merely wishes to check on message status. Even if email is automatically sent to alphanumeric pagers (this is done at the Media Lab and many other locations), each page is limited to several hundred characters; this asynchronous notification provides even greater motivation to immediately call in from the nearest telephone to hear the entire body of an urgent message and possibly send a timely reply.

SUMMARY

This chapter has discussed the advantages and disadvantages of voice from the perspective of interactive systems employing voice response. Speech output may be difficult to employ effectively for information providing because it is slow, temporal, serial, "bulky," and may not be private enough for some office situations. At the same time speech is advantageous over other media because it can be used at a distance over the telephone or heard across a room or when a user's hands and eyes are otherwise occupied.

A number of design considerations were suggested based on comparing the advantages to the disadvantages of speech output and how it should be employed in an application. Some applications may simply be inappropriate for speech output. For appropriate applications, speech may be sped up to increase performance, but a means of repeating an utterance may be needed to clarify messages when a user fails to understand them. There may be a role for multiple voice styles in a single speech-output application.

Because many interactive voice response systems are telephone based, we paid particular attention to the issue of user input with the telephone keypad. Touch tones can be used to select an item from a list, to answer yes-or-no questions, or to make a selection from a menu. Implicit or explicit termination of input may be more suitable to different tasks being performed over the telephone.

Three case studies were presented to illustrate the points being made in this chapter. Direction Assistance employs a variety of input techniques to trigger discourse generation from a complicated computer database of street segments. Back Seat Driver, using the same database, must generate directions in real time

and instill confidence in the user that it remains on target despite periods of silence—a need not present in graphical interfaces. Finally, Voiced Mail illustrated how text filtering and organizational issues can minimize the amount of information that must be attended to while reading electronic mail with speech synthesis; it also eschewed prompts and menus in favor of a default flow of control during a session with a caller.

Speech Recognition

This chapter is about the technologies used to allow computers to recognize the words in human speech. It describes the basic components of all speech recognition systems and illustrates these with an example of a simple recognizer typical of several inexpensive commercial products. After this discussion of basic recognition, the chapter details a larger range of features that can be used to differentiate various recognizers according to the styles of speech interactions they support. Several more advanced recognition techniques are then introduced followed by brief descriptions of selected research projects in large vocabulary and speaker independent word recognition.

BASIC RECOGNIZER COMPONENTS

There are three basic components of any speech recognizer.

1. A speech **representation** that is computationally efficient for pattern matching. The representation is the form into which the recognizer converts the speech signal before it begins analysis to identify words. Typical representations include the output of a bank of filters (similar to a spectrogram), Linear Predictive Coding (LPC) coefficients,¹ and zero crossings of the speech waveform. Recognizers of

¹See Chapter 3.

increased sophistication incorporate more abstract representations of speech such as phonemes or distinctive spectral features. Hidden Markov Models, described later in this chapter, are a statistical representation based on the various ways words or phonemes may be pronounced.

2. A set of **templates** or **models**, which are descriptions of each word to be recognized, in the representation of speech used by the recognizer. The templates describe the words in the recognizer's vocabulary, i.e., those words that the recognizer can identify. They are reference models against which an input can be compared to determine what was spoken.
3. A **pattern matching** algorithm to determine which template is most similar to a specimen of speech input. This element of the speech recognizer must determine word boundaries, locate the most similar template, and decide whether the difference between the input and the selected template is minor enough to accept the word. In very large vocabulary recognizers, the pattern matching technique usually includes access to more advanced knowledge of language such as syntax and semantics, which constrain how words can be combined into sentences.

To identify a word the recognizer must capture incoming speech and convert it to the chosen internal representation (see Figure 7.1). The pattern matcher then selects the template that most closely matches the input or rejects the utterance if no template is a close enough match.

SIMPLE RECOGNIZER

This section describes a simple recognizer typical of many inexpensive, commercially available devices. This recognizer is designed to identify a small number of

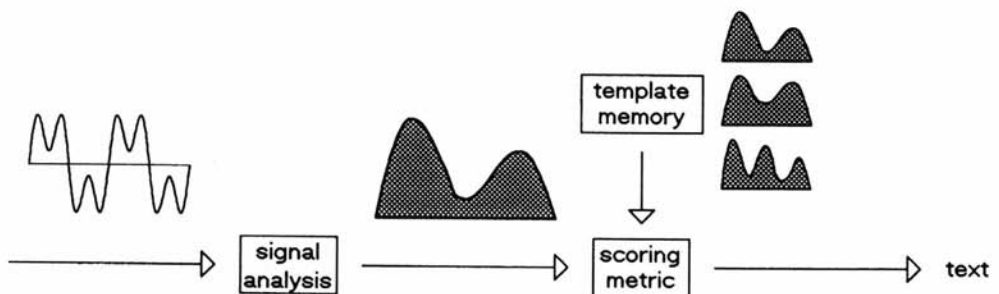


Figure 7.1. The functional elements of a speech recognizer. The user's speech is digitized and converted to the recognizer's internal representation. The captured speech is then compared with the words stored in the recognizer's template memory. The pattern matching algorithm then determines which template, if any, is the closest match.

words spoken in isolation by a specific individual. The purpose of the description that follows is not to provide details of a particular product but rather to offer a sample implementation of the basic components described previously.

Representation

A simple recognizer digitizes incoming audio by means of a codec² and then uses a digital signal processing algorithm to extract frames of Linear Predictive Coding (LPC) coefficients every 20 milliseconds. The LPC coefficients are a concise representation of the articulatory characteristics of speech as they capture both the resonances of the vocal tract as well as its voicing characteristics. The 20 millisecond sampling interval results in 50 LPC frames per second, providing significant data reduction to simplify later pattern-matching.

Templates

Templates are gathered by prompting the user from a word list and then saving a set of LPC frames for each word generated as just described. To build a template, the recognizer must determine when the user begins and finishes speaking to know which LPC frames to include in the template. Since each LPC frame or set of coefficients in one 20 millisecond sampling period includes an energy value, this task can be accomplished easily. As shown in Figure 7.2, once the audio exceeds the threshold of background noise, LPC frames are saved until the audio drops below the threshold. It is necessary to wait until the signal has dropped below the threshold for a short period of time as silence or low energy can occur within a word, such as at stop consonants.

In the case of this simple recognizer, templates are trained by a user saying each word once. More sophisticated recognizers may take multiple specimens of a

²See Chapter 3.

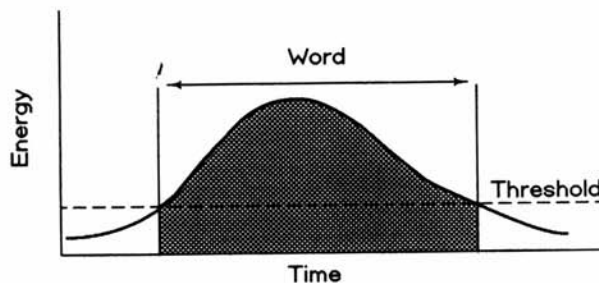


Figure 7.2. Energy thresholds can be used to find word boundaries if the words are spoken in isolation.

word to build a more robust template. Once templates are trained successfully, the user can save them as a disk file on the host computer and reload them before each session, to avoid template training each time the recognizer is used.

This recognizer also allows retraining of a single template, in which case the new LPC frames replace the old. Retraining is useful if the vocabulary changes, if the original training was poor, or if the user decides that a different pronunciation of the word is more comfortable. Retraining may also be necessary when the user has a cold, due to an obstructed nasal cavity.

Pattern Matching

Word recognition requires the detection and classification of speech input. Just as with template creation, a word is captured by detecting that the audio input exceeds the background noise threshold and then converting the input to LPC frames until the input again drops to the background level. The input word is then compared with each template; each frame or set of LPC parameters of the input is compared to the corresponding frame of the template. The frame-by-frame error is the sum of the differences between each of the dozen or so LPC parameters; the error for the word is the sum of the errors for each frame. The template with the smallest word error most closely matches the audio input. If the error exceeds a rejection threshold, a failed recognition attempt is reported. Otherwise the recognizer reports the word corresponding to the closest template.

Both the templates and the captured audio input are multidimensional vectors, with one degree-of-freedom per LPC parameter and one for time. But to illustrate pattern matching let us assume that the templates consist of only a pair of values that display on two axes. One parameter is assigned to the X-axis and the other parameter maps to the Y-axis so that each template occupies a point whose coordinates are the two parameters (see Figure 7.3). Speech input is converted into this representation, i.e., a pair of parameters specifying a point in the same parameter space as the templates. The closest template is the nearest neighbor template to the point representing the input. The four templates divide the space into four regions; any pair of parameters in each region most closely matches the template in that region.

Even the simplest of speech recognizers can improve its accuracy by employing a better pattern matching algorithm. Two additional refinements, illustrated in Figure 7.4, contribute to this recognition improvement. The first refinement is that the input cannot exceed a threshold distance (r in the figure) from the nearest template. When the audio input is further from the nearest template than this distance, a rejection error is reported. This is necessary to avoid inadvertently accepting noise, breath sounds, or a mistaken utterance as one of the acceptable words in the recognizer's vocabulary.

The second refinement is a requirement that the input word maps significantly closer to the best match template than any other. If a point is very nearly the same distance from two or more templates, the recognizer is confident that one of these was spoken but uncertain which particular word was spoken. This is depicted in Figure 7.4 as the shaded regions around the lines partitioning the

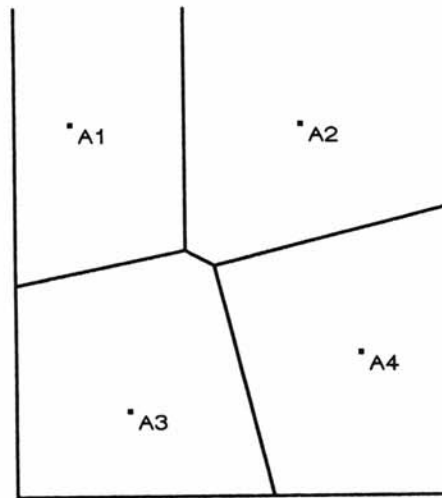


Figure 7.3. To illustrate pattern matching decisions, assume that each template represents two degrees of freedom defining a Cartesian coordinate space. Four templates occupy four points in this space.

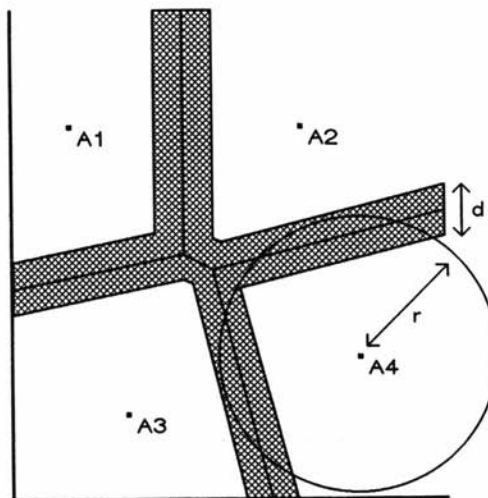


Figure 7.4. A more refined classifier has upper bounds r on the distance between an input specimen and a chosen template. It will also reject values which are closer to one template but also close to another, i.e., within distance d .

space between neighbors; the distance d defines the minimal difference required to differentiate candidate choices. Points within the shaded region are nearly the same distance from each of the two closest templates and will also cause a rejection error.

This section described an extremely simple recognizer that identifies only isolated words. The speech recognizer's simplicity is due in part to its unsophisticated processing of acoustical information: It does not extract significant features but merely considers LPC frames in isolation. Because each template may contain many frames, the pattern matcher has to make a large number of frame-by-frame difference calculations, making it computationally difficult to recognize a large number of words. More sophisticated recognizers detect and classify key acoustic features of an utterance and use this for the internal representation.

Additionally, the pattern matcher just described makes no provision for templates of different lengths and, more importantly, assumes that the input utterance will be of the same length as the correct template. If acoustic boundary detection operates slightly differently from word to word or if the words are spoken at different speeds, this pattern matcher will likely fail. A technique known as **dynamic time warping** or **dynamic programming** (discussed later in this chapter) is often used to compensate for time differences.

CLASSES OF RECOGNIZERS

Recognizers vary widely in their functionality. In addition to performance, other distinctions between recognition techniques are important in selecting a recognizer for a particular application. These distinctions are summarized in Figure 7.5, which displays a three-dimensional space representing the range of possible recognizer configurations. Some of these recognizers are commercially available and have been in use for some time, whereas others are the subject of ongoing research and are just beginning to emerge from research laboratories.

Who Can Use the Recognizer?

A **speaker-independent** recognizer is designed to recognize anyone, while a **speaker-dependent** recognizer can be expected to understand only a single particular speaker. A **speaker-adaptive** recognizer functions to an extent as a combination of the two; it accommodates a new user without requiring that the user train every word in the vocabulary.

Speaker-independent recognition is more difficult than speaker-dependent recognition because we all speak the same words slightly differently. Although we usually understand each other, this variability in pronunciation and voice quality among talkers plays havoc with the pattern matching algorithms of simple recognizers. Speaker-independent recognition requires more elaborate template generation and a clustering technique to identify various ways a particular word may be pronounced.

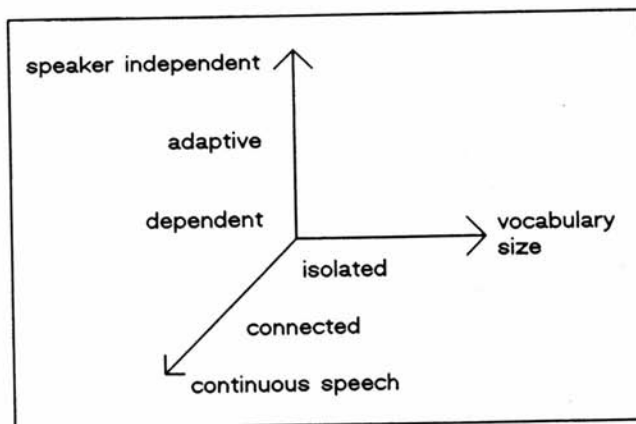


Figure 7.5. A three-dimensional space defined by the different functionalities provided by a recognizer.

This process of classifying the acceptable variations in word pronunciations is the equivalent of the training method just described for the simple illustrative recognizer. Since speaker-independent recognition is such a difficult task, most speaker-independent recognizers are limited to small vocabularies (10 to 20 words) and recognize only single words spoken in isolation. Several commercially available speaker-independent recognizers accept the digits “0” through “9” plus “yes” and “no.” Speaker-independent recognition for a larger vocabulary or with words spoken together must rely on additional constraints such as a heavily restricted grammar to achieve adequate recognition performance.

An individual pronounces a single word in a much more consistent fashion than that of a number of different people; this is why speaker-dependent recognition is less difficult. Most speaker-dependent recognizers use a training method similar to the one described for the simple recognizer in the previous section. Multiple templates may be trained and merged together to give an average pronunciation for each word in the vocabulary.

A speaker-adaptive recognizer learns to adapt to a particular speaker either by calibrating itself to a known sample of that person’s speech or by incorporating user feedback when it recognizes a word correctly or incorrectly. Such a recognizer has a basic acoustic model for each word that it refines to fit a particular speaker. One common technique is to have the user read a passage of training data; the recognizer can then determine how this particular person speaks.

If a large vocabulary size is needed, a recognizer must be either speaker-adaptive or independent because training many templates is tedious and time consuming. Some adaptive recognizers require many hours of computer time to derive the user model from the training data, but once the data has been acquired the user is freed from the task. The resulting user model can be saved in a host computer and downloaded later, similar to the set of word templates for the speaker-dependent recognizer.

Speaking Style: Connected or Isolated Words?

A **discrete speech** recognizer requires pauses between each word to identify word boundaries. A **connected speech** recognizer is designed to recognize a short series of words spoken together as a phrase. A **continuous speech** recognizer is capable of identifying words in a long string of ordinary speech without the talker pausing between groups of words. A **keyword spotting** recognizer can locate a few words in the midst of any amount of speech.

“Connected speech” for recognition purposes is not natural speech. Users of a connected speech recognizer can speak a few words or perhaps a whole sentence at a time but then must pause to let the recognizer catch up. Users must also speak distinctly. Pausing after each sentence provides reliable boundary points for the first and last word, which facilitates recognition of the entire phrase. But we do not pause between sentences in fluent speech. Human listeners can keep up, and this is the goal of continuous speech recognition, which may become a reality as computer speeds increase.

Keyword spotting searches for a small number of words in a stream of continuous speech. For example, a keyword recognizer might be designed to recognize the digits from an utterance such as “The number is three five seven . . . um . . . four one, please.” Successful keyword spotting is comparatively new [Wilpon *et al.* 1990] and more difficult than simply separating speech from non-speech background noise.

Most currently-available commercial products recognize isolated or connected speech. Connected speech is much faster to use because it eliminates the need for unnatural pauses and requires less attention to speaking style on the part of the user. But because connected recognition is more difficult, it may manifest higher error rates, which could outweigh the speed advantage. It can also be argued that speaking discretely is more effective because it requires the user to keep in mind the need to speak clearly from a limited vocabulary while using speech recognition [Biermann *et al.* 1985].

Connected word recognition is much more difficult than isolated word recognition for several reasons.

- Coarticulation changes the pronunciation of a word as a function of its neighbors. Initial and final syllables are particularly subject to modification. Words spoken in isolation do not suffer from coarticulation effects between words.
- It is difficult to find word boundaries reliably from within fluent speech. There is no pause between words, nor is there a significant decrease in speech energy at word boundaries; low energy syllables containing stop consonants are often more discernible than word boundaries.
- The probability of error increases with the number of words in an utterance. If the first word is incorrectly matched against a template which is either too long or too short, then the data to be analyzed for the second word will be incorrect, making the error propagate to subsequent words.

Because of these factors many current applications of speech recognition are based on isolated word recognizers.

Vocabulary Size

Another criterion by which to differentiate recognizers is vocabulary size, which can be grossly categorized as small, medium, or large. Small vocabulary recognizers with less than 200 words have been available for some time. Medium size recognizers (200 to 5000 words) are being developed, usually based on the same algorithms used for smaller vocabulary systems but running on faster hardware. Large vocabulary recognizers aim for the 5000 to 10,000 word level. Much ordinary office language could be handled with a vocabulary of this breadth, which marks the range being aimed for in "listening typewriters" designed to accommodate dictation of business correspondence.

Several issues conspire to complicate the recognition of large vocabularies. One factor is the computational power required for the pattern matching algorithm. The input must be compared with each template, so classification time is a function of the number of templates, i.e., vocabulary size. The requirement of an acceptable response time therefore puts an upper limit on vocabulary size. As microprocessor speeds increase, this computational limit will become less of an issue. Some search-pruning techniques can also be employed to quickly remove candidate templates from the search if they are obviously poor choices, such as if they are much too long or short. If the recognizer employs a grammar, it can use this knowledge of how words can be combined to eliminate syntactically incorrect candidates at each point in the search.

A more serious limitation to vocabulary size is simply that as the number of words increases, it is more likely that the recognizer will find some of them similar. To return to the two parameter template model discussed earlier, compare Figure 7.3 with Figure 7.6. As the number of templates increases, the average distance between them decreases, allowing for a smaller margin of difference in pronunciation between the input sample and its associated template.

Decreased distance between templates is confounded by the fact that as templates or words are added to the recognizer's vocabulary, they are not uniformly distributed throughout the recognizer's representation space. In fact, some words in the vocabulary are likely to sound so similar that to distinguish among them on acoustic evidence alone is extremely difficult. For example, if a vocabulary consisted of the words, "sun," "moon," and "stars," we might expect that distinguishing which word was spoken to be easy. But if we add "Venus," "Earth," and "Mars" to the vocabulary, we might anticipate confusion between "stars" and "Mars."³

³This is a simplistic example. The strong fricative and stop at the beginning of "stars" should be easily differentiated from the nasal of "Mars" if suitable acoustic features are used for the recognizer's representation. It is hard to predict which words will sound similar without intimate knowledge of the recognizer's internal speech representation.

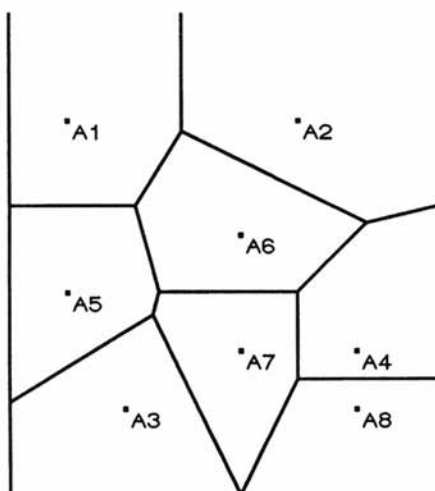


Figure 7.6. As the number of words increases, the mean distance between templates decreases.

ADVANCED RECOGNITION TECHNIQUES

This section explores several techniques to enhance speech recognition. The simplest recognizers, supporting a very small speaker-dependent vocabulary of isolated words, occupy the region near the origin of the three-dimensional model depicted earlier in Figure 7.5. In moving away from the origin in any direction, recognition errors are more likely as the process becomes more complicated. But a large vocabulary, continuous speech, and speaker independence are precisely those attributes that make recognition more widely useful.

Unless users can be convinced to change their habits to speak more clearly and consistently, recognizers must be improved. More advanced recognizers generally employ one of two techniques to make pattern matching more powerful for dealing with variations in speech patterns. These techniques, **Dynamic Time Warping** and **Hidden Markov Models**, are the topics of the next two sections. The descriptions that follow provide an overview of the two techniques; the curious reader is encouraged to consult the references for a more rigorous description of the algorithms. A third approach to managing speech pattern variation uses **neural networks** for speech recognition; this approach is far less developed than the previous two and still speculative.

Data reduction can facilitate pattern matching by minimizing its computational requirements. **Vector Quantization** is a technique employed by many recognizers to capture important variations in speech parameters without overloading the classifier—so it will be described in this section as well. Finally, the last part of this section considers how nonspeech evidence could be used to improve large vocabulary speech recognition.

Dynamic Time Warping

Dynamic Time Warping is a technique that compensates for variability in the rate at which words are spoken. Dynamic Time Warping (DTW) was developed primarily as a mechanism to compensate for variable word duration in connected speech recognition [Sakoe and Chiba 1978]. This method can also help determine word boundaries when an unknown number of words are spoken together. DTW is based on a more general computational technique known as **dynamic programming**.

The duration of spoken words is quite variable, especially if we compare connected speech with the isolated words that may have been used for training a recognizer. Coarticulation may shorten words by combining their boundary syllables with the preceding or subsequent words. Words spoken in isolation (citation form) are longer and often more clearly enunciated. The stress pattern of the sentence lengthens some syllables, as stressed syllables are longer than unstressed syllables. Such changes in length are not linear; every phoneme is not lengthened by the same scale factor. DTW compensates for such nonlinear phenomena.

Consider the simple recognizer discussed earlier in this chapter. It computes an error between the input speech and a template by computing the frame-by-frame difference between the two. But if the talker speaks at a different rate, successive frames of input may not align with the same phoneme in the template, giving a deceptively large error. For example, if the word "fast" is trained but the talker lengthens the vowel ("faaast") during recognition, some frames of the lengthened vowel would be scored against the frames corresponding to the unvoiced "s" in the template. DTW detects that successive "a" frames of the input match the fewer "a" frames of the template better than the "s" frames that follow, and it computes an error based on the selected matchup.

DTW operates by selecting which frames of the reference template best match each frame of the input such that the resulting error between them is minimized. By allowing multiple frames of one to be matched against a single repeated frame of the other, DTW can compress or expand relative time. Because this decision is made on a frame-by-frame basis, the time-scaling is local; DTW may compress one portion of the input and expand another if necessary.

DTW provides a mapping between the sample and a reference template such that the error when the two are compared is minimized. This mapping defines a path between sample and reference frames; for each frame of the sample it specifies the template frame best matched to the next sample frame. In Figure 7.7, the n th sample frame is compared to the m th reference frame. If the sample is spoken more quickly, then multiple reference frames correspond to a single template frame; this case is indicated by the vertical path, which allows the n th sample frame to also match the $m + 1$ st reference frame. If the reference and sample are proceeding at the same rate, the $m + 1$ st reference frame will match the $n + 1$ st sample frame forming a path at a 45-degree angle (slope = 1). Finally, if the sample is spoken more slowly than the reference, multiple sample frames must be compared to a single reference frame as indicated by the horizontal line.

Figure 7.8 shows how these frame-by-frame decisions combine to produce a path for comparing the sample to the reference. The DTW algorithm computes

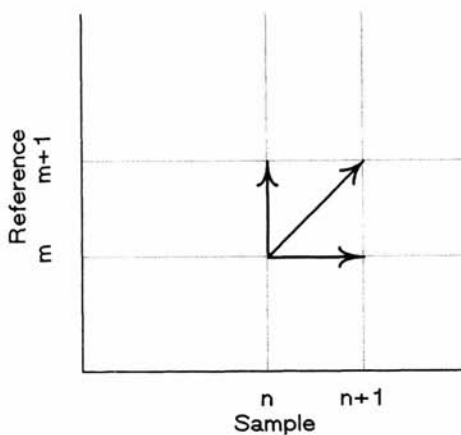


Figure 7.7. Dynamic Time Warping defines a path between frames of a sample utterance and a template such that the frame-by-frame error between the two is minimized. If sample point m matches reference point m , then reference point $m + 1$ may match either sample point n or $n + 1$.

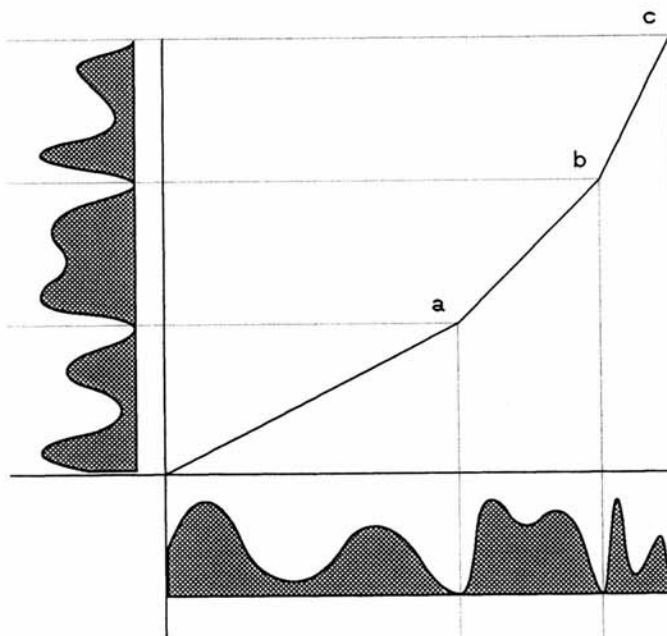


Figure 7.8. Dynamic Time Warping provides nonlinear time-scaling between a sample (horizontal-axis) and reference (vertical-axis) utterances. In region **a** the sample is spoken more slowly; in region **b** they are spoken at the same rate; and in region **c** the template is spoken more slowly.

the minimal error path between the input and template frames; relative speaking rate can be inferred from the slope of this path. A slope of 1 (45-degree line) is optimal when both are spoken at the identical rate.

It is necessary to place limits on the degree of time-scaling allowed by DTW. It would be senseless to compare 50 milliseconds of the sample to 1200 milliseconds of reference; normal speech does not exhibit such wide variability. Constraints on the maximum time-warping ratio are imposed by placing upper and lower bounds on the slope of the DTW path. The typical range of slopes is one-half to two, which accommodates a sample being spoken either half or twice as fast as the reference.

“Two-level” dynamic programming can be used to find word boundaries in a connected utterance containing an unknown number of words. At the lower level, DTW allows time-scaling of each word by computing the minimal error path matching the input to each template. At the higher level, these word-by-word error values are used to compute the best path through the set of all possible word combinations that could comprise a sentence. The two levels operate in concert to apportion the duration of the utterance among the set of selected templates. Figure 7.9 shows how a number of words of different lengths could consist of similar phones, or speech sounds.⁴ It may be that the first three phones most closely match the short word “her,” which would suggest that the utterance contained three words. But a better *overall* score might be obtained by matching these three phones against the beginning of “heartfelt” and selecting the utterance with only two words.

Dynamic programming is most beneficial for connected speech recognition. Word durations show greater temporal variation in connected speech than in isolated speech. Additionally, word boundaries must be found by considering how the continuous stream of phones can best be segmented into words in the absence of interword pauses. Dynamic programming is used in a number of commercial connected speech recognition systems, but its popularity for research purposes has faded in favor of Hidden Markov Models.

Hidden Markov Models

A **Hidden Markov Model (HMM)** is a two-stage probabilistic process that can be used as a powerful representation for speech. A Hidden Markov Model is a well-behaved mathematical construct, and a number of detailed algorithms exist for solving problems associated with HMMs; introductions to these can be found in [Rabiner and Juang 1986a]. This section first explains HMMs in the abstract and then demonstrates how they can be applied to speech recognition.

An HMM consists of a number of internal states; the model passes from an initial state to the final state as a step-by-step process generating an observable output at each step (state transition). For example, the states may correspond to phonemes contained in a word with the observable output corresponding to the

⁴Such a representation is called a **word lattice**.

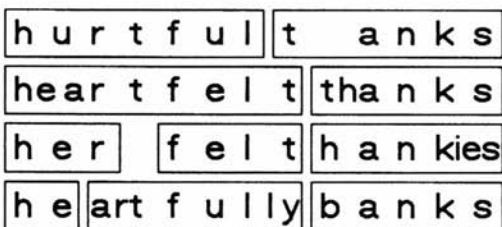


Figure 7.9. Different sequences of words could match an utterance; each sequence implies different word boundaries in the input utterance.

presence or absence of a number of acoustic features. At each step, the model can either move to a new state or stay in the current one. The model is “hidden” in that we cannot observe the state directly but only its output. From the series of observable outputs, we can attempt to guess when the model was in each state. Alternatively, we can say whether some series of outputs was likely to have been generated by a particular HMM.

For example, consider the arrangement shown in Figure 7.10, in which a box and a bowl are each full of black balls and white balls. The box has many more black balls than white while white balls dominate the bowl. Starting with the box, we remove one ball from it at random and pass it to an observer in another room who cannot see what is being done. Then a normal six-sided die is tossed. If the result is less than four, then the source of the next ball is the bowl. If the die is greater than three, then the next ball will be selected from the box. The cycle is then repeated.⁵ Whenever we select from the box, a die throw of less than four shifts attention to the bowl. Once we start selecting from the bowl, we continue selecting from it unless the die shows a one, at which point we are finished. This model is likely to spend a majority of its cycles selecting from the bowl since a state transition from the box has a probability $\frac{1}{2}$ of shifting to the bowl, but a bowl transition has a probability of only $\frac{1}{6}$ of terminating and $\frac{5}{6}$ of continuing with the bowl.⁶ We should expect this arrangement to initially present more black balls than white to the observer (while it is in the box state) and then to produce more white balls. We should also expect more white balls overall. But keep in mind that this is a probabilistic process: Not all black balls come from the box and not all selections come from the bowl.

The box-and-bowl setup is a Hidden Markov Model; it is characterized by a number of *states* and associated probabilities for each *transition* from any state. The box and bowl are each a state; the rules about throwing the die define the

⁵This pedagogic example was inspired by Rabiner and Juang [Rabiner and Juang 1986b].

⁶Probability is expressed as a number between zero and one. If an event occurs with probability $\frac{1}{2}$, we would expect it to happen once in two trials. A probability of $\frac{1}{6}$ implies that we can expect the event once out of six trials. A smaller probability indicates that an event is less likely to occur.

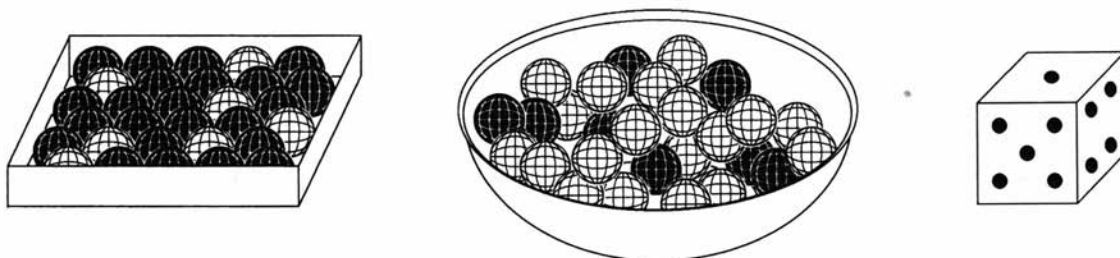


Figure 7.10. A box and a bowl full of colored balls illustrate a Hidden Markov Model.

transition properties. Figure 7.11 shows a more general HMM; in each state $S(n)$ there is some probability $P(n, n)$ of remaining in that state and some probability $P(n, n + 1)$ of transitioning to the next state. In some models, it may be possible to bypass a state or a series of states as indicated by the dashed arc labeled P_{13} in the figure. The sum of the probabilities of all the arcs leaving a state is one; the model accounts for every possible path through its states.

How does this apply to speech recognition? An HMM can be used to represent a word with internal states representing characteristic acoustic segments, possibly phonemes or allophones. The output of a state is a frame or vector of acoustic parameters or features; this output is probabilistic to allow for variability in pronunciation and hence differences in the acoustic representation. The duration of an acoustic segment is a function of the number of steps in which the model is in the state corresponding to the segment. Staying in the same state, i.e. lengthening a phone, depends on the probability associated with the transition from that state to itself (P_{11} in Figure 7.11). Arcs such as P_{13} may be included to indicate that an intermediate state $S2$ is optional during pronunciation, such as the second syllable in “chocolate.”

In terms of the basic recognizer components, the “templates” consist of a set of HMMs with one HMM associated with every word. The acoustic representation of an input specimen of speech is *not* a collection of states but rather a set of acous-

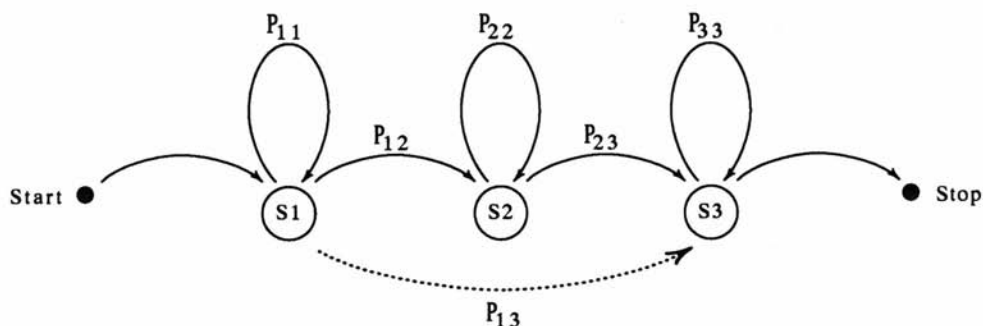


Figure 7.11. A Markov Model.

tic parameters corresponding to the possible observed sequences from the template HMMs. For the HMM-based recognizer, classification consists of determining which HMM has the highest probability of producing the observed acoustic sequence. The Viterbi algorithm is a commonly used method of solving this classification problem; a more detailed description is beyond the scope of this book but see [Lee and Alleva 1992] for an overview of its theory.

There are two related HMM problems which can also be solved mathematically. One is training the statistically defined HMMs that the recognizer requires as templates. Given some number of observation sequences (i.e., repetitions of the word), an HMM must be derived to represent that particular word. The second problem, not directly relevant during recognition, is to determine for some HMM and a set of observed outputs the internal states of the model that were most likely at each step.

Hidden Markov Models are a powerful representation of speech and consequently are used in many speech recognition systems currently under development. Although this section has described isolated word recognition, HMMs can also be used with connected speech to represent not only the phones of each word but also the probabilities of transitioning from one word to another. For connected speech recognition, the observed sequence would be generated by passing through a sequence of states in a sequence of HMMs with one HMM for each word spoken. The combination of words is also described statistically as another layer of HMMs in which each state corresponds to an entire word thereby encoding syntactic and semantic information.

Vector Quantization

Vector Quantization (VQ) is a technique employed for data reduction in speech coding and speech recognition. VQ is used to reduce a widely ranging input value into a smaller set of numbers representing typical values of the input. Because the input is classified as one of a small number of values instead of the value itself, less storage space is required. More importantly, the input is usually multidimensional (e.g., a set of LPC coefficients), and it is reduced to the single dimension identifying the nearest vector-quantized value.

In order to perform Vector Quantization, it is first necessary to decide how to efficiently cluster all possible input values. The analysis of sample data, which must be typical of the data to be encoded, determines how to cluster similar input values. The cluster's "center of gravity," or average value, then represents the entire set of data in the group. Each cluster is represented as one entry in a **codebook**, which lists the average value of each group. Having created a robust VQ codebook from specimen data, new data can be classified by selecting the codebook entry nearest in value to that data. The original value of the input data is lost; it is now represented by the single value associated with the codebook entry.

Building the codebook divides the space of possible values into regions and *typical* values. This is much like the process of selecting word templates for recognition as shown in Figure 7.6. The number of entries required for the codebook

depends on how well the data clusters into groups and the accuracy with which the data must be represented. If the data points are widely scattered, it is difficult to cluster them efficiently. The range of values subsumed by each cluster determines the accuracy with which vector quantized data can be represented since the codebook entry is used to look up a value for the data and only one value is stored per cluster.

Representing an input as a codebook entry having a similar value significantly reduces both storage and computation during pattern matching. To vector quantize an input value based on a codebook of N entries, the input must be compared with at most N possible values. The error between each of the codebook values can be computed in advance and stored in a two-dimensional array. During pattern matching the error between an input value and a corresponding template value can be found by table lookup instead of direct computation. The sum of the errors for each frame is the error between the input and that template.

As a simple example, vector quantization can be used to represent the number of hours per day that light bulbs are left burning in a home. If daily use is represented as an integral number of hours from zero to twenty-four, five bits of storage are required for each bulb. Higher resolution, such as minutes in addition to hours, requires even more bits of storage per bulb. However, by observing the individual bulbs we discover some interesting patterns of usage. Many bulbs are rarely turned on. Some bulbs are always left on all night like an outdoor lamp or a child's night light. Other bulbs, in the living room and kitchen, may typically be on from dusk until the occupants go to sleep. Still other bulbs such as in a closet may be turned on occasionally for brief periods of a few minutes each day.

This situation can be described by a vector quantization scheme employing a codebook with four entries to cover the four typical cases described in the previous paragraph.⁷ Each entry in the codebook (see Figure 7.12) corresponds to one case of light bulb use, and the value stored for that entry indicates how long such a bulb is likely to be used. Using this codebook, light bulb use is encoded in two bits at a savings of three bits per bulb.

Vector quantization is usually employed to represent multidimensional quantities so the compression to a single-dimensioned codebook offers further data compression. To continue with the light bulb example, we might observe that many of the lamps used rarely or left on all night are low wattage, e.g., 60 or 75 watts. The living room and kitchen lights are more likely to be 100 or 150 watts to illuminate work areas. So we might vector quantize this information as well as shown in Figure 7.13. This new codebook indicates that a bulb assigned value two is typically 120 watts and on for five hours a day. Note that a 150 watt bulb used seven hours a day and a 75 watt bulb used four hours a day will both be represented by codebook value 2 as well. It may be that there is actually no single bulb of 120 watts; this is an average that represents the least error when compared to all the bulbs in the sample set with this codebook value.

⁷Such a codebook is said to be "of size four."

codebook entry	codebook value
0	0 hours
1	15 minutes
2	5 hours
3	10 hours

Figure 7.12. A simple VQ codebook representing light bulb use.

Once quantized, detailed information about a particular light bulb is lost, but we still have a general idea of how much each is used. This would be useful for identifying which bulbs should be replaced by more expensive energy-efficient models. While the memory savings are small in this example, vector quantization can be used for much greater data reduction in speech coding applications where it represents complex acoustical information about an utterance.

How is vector quantization used in speech representation? Consider the output of an LPC coder; it includes precise indications of the locations of each of the formants, or resonances, in the vocal tract as conveyed by filter parameters. But for gross categorization of the position of the vocal tract, e.g., to differentiate the vowels by their F1/F2 ratio, such precision is not necessary. Furthermore, not all combinations of F1 and F2 are possible since they are constrained by the physical characteristics of the vocal tract. Vector quantizing these filter coefficients can capture the overall vocal tract configuration in a very concise form when great precision of each parameter is not necessary.

Employing Constraints

A recognizer's task becomes increasingly difficult as the number of words to be recognized increases. To improve recognition large vocabulary systems apply constraints to the set of all possible words that can make up an input utterance.

codebook entry	codebook value	
	time	wattage
0	0 hours	65
1	15 minutes	75
2	5 hours	120
3	10 hours	70

Figure 7.13. Vector quantization is usually used to represent multidimensional quantities. Here it encodes both the average duration of illumination as well as the wattage of bulbs.

Constraints limit the search space during word classification by ruling out improper combinations of words. Although we readily employ general syntactic and semantic constraints to rule out many ill-formed or nonsense sentences while trying to understand a talker, it is difficult to express the knowledge that underlies such decisions in computationally tractable terms. Current recognition systems incorporate limited constraints specific to a given task.

Some of these constraints are specific to the the input; for example, North American telephone numbers have seven digits (ten if the area code is included). Other sets of constraints can be derived from a specification of all the legal utterances for a set of words and a task. For example, simple rules of syntax tell us that every sentence must have a verb, and common sense dictates that an application for ordering pizza should expect to encounter the word “large” adjacent to “pepperoni.” To define these constraints the application designer must specify every legal sentence to be recognized. This is facilitated by lumping groups of words into classes which have meaning in a particular task context, such as “digit,” “size,” or “topping.” Sentences can then be described as series of words combined from particular classes instead of listing every possible word sequence.

Because they are derived by observing people perform a task and computing the likelihood of particular sequences of words, constraints can also be probabilistic. This requires a large body of data from which to derive the probabilities as there may be many possible sentences and statistical representations that require a large quantity of training data. Some language models have been based on analysis of a large corpus of text documents. Although spoken and written language differ, the advantage of this approach is that the text is both readily available and easily analyzed by a computer.

The effectiveness of constraints is related to the degree to which they limit the possible input utterances; this can be stated as the predictive ability of a string of n words on the $n + 1$ st word. Alternatively, the constraints may be quantized as the **branching factor** associated with any node in a string of input words; this number indicates how many different words can follow one or more previous words. A question requiring a yes-or-no answer has a branching factor of two, for example, while one requiring a single digit has a branching factor of ten. **Perplexity** is a measure of the branching factor averaged across all possible word junctures in the set of legal utterances. The lower the perplexity, the more effectively lexical constraints can improve recognition.

Recognizers based on either *ad hoc* or statistical models of language and the task at hand may be subject to limited portability. The words a talker uses change as a function of the topic of discourse, and a recognizer may not be able to make such a shift. For example, a carefully constructed grammar for a task that allows the user to make inquiries into the state of ships in a navy’s fleet will be rather different from one used in general business correspondence. Recognizers do not yet make use of language representations general enough to make specification of the task a simple problem. A more general discussion of methods of specifying both syntactic and semantic constraints is found in Chapter 9.

ADVANCED RECOGNITION SYSTEMS

This section briefly describes details of three advanced speech recognizers that are considerably more sophisticated than the simplistic example presented early in this chapter. These recognizers are research projects, not yet commercial products. This is hardly an exhaustive list of speech recognition research, but it is intended to be representative of the approaches being used in modern, large vocabulary recognition systems. These descriptions will almost certainly be out of date by the time this book is actually published so they should be taken as little more than a snapshot of a rapidly evolving field.

IBM's Tangora

IBM's Tangora speech recognizer is designed to recognize 5000 to 20,000 words of isolated speech [Jelinek 1985]. A real-time version of this research project was implemented on a single-slot personal computer board. The Tangora recognizer is speaker adaptive with training based on the user's reading a few paragraphs of text to calibrate its acoustic model.

The Tangora recognizer uses a vector quantization front end to classify acoustic input, which is then matched to words using discrete Hidden Markov Models of phoneme realization. A linguistic decoder then classifies the output of the acoustic front end producing a word for output. Tangora is targetted at automatic transcription of speech to text in a dictation context.

The linguistic constraints employed by the second stage of this recognizer are based on the probabilities of groups of two or three words occurring in sequence. This statistical model, called a **bigram** or **trigram grammar**, was derived through analysis of a large quantity of text from business correspondence. It should be readily apparent that such a representation of the language includes both syntactic and semantic information as well as indirectly encoded world knowledge. Not only is "late I slept" an unusual sequence due to constraints, but, similarly "I slept furiously" is improbable due to its ill-formed semantics; "the green cat" is unlikely due to world knowledge that cats do not have green fur. But this model cannot distinguish the source of knowledge; it merely represents the end result of all factors that contribute to likely word sequences.

CMU's Sphinx

Carnegie-Mellon University's Sphinx recognizer is designed for speaker-independent connected speech recognition [Lee and Hon 1988, Lee 1988]. At the time of this writing, it operates in nearly real time using hardware multi-processor peripherals to aid in the search process. Sphinx operates with a 1000 word vocabulary and achieves a recognition accuracy percentage in the mid-nineties when provided with a bigram language model; however, accuracy drops to the mid-fifties without the grammar model.

Sphinx makes extensive use of Hidden Markov Models: each phone is represented by an HMM, each word is a network of phones, and the language is a network of words. Phones in function words are modeled separately from other phones because of the higher degree of coarticulation and distortion in function words. Because function words are common to all vocabularies, they can be modeled using more thoroughly trained HMMs.

Sphinx uses multiple codebook vector quantization as its acoustic front end. Separate codebooks are used to encode energy, cepstral, and differential cepstral parameters. The cepstrum is a signal analysis construct gaining popularity with speech recognition because it differentiates voicing and vocal tract aspects of the speech signal. Differential cepstrum measurements indicate how the cepstrum changes from one sampling period to the next; this measure is particularly useful for analysis of the most dynamic aspects of speech such as many of the consonants.

MIT's SUMMIT

The SUMMIT [Zue *et al.* 1989a] system is a phoneme-based connected speech recognizer being developed by the Spoken Language Systems Group at M.I.T. In recognizing speech, SUMMIT first transforms the speech signal into a representation modeled after the auditory processing that occurs in our ears. This is a non-linear transformation based on Seneff's auditory model [Seneff 1988]; it enhances acoustic information crucial for recognizing speech and suppresses irrelevant detail. Part of this analysis models the transduction between the hairs in the cochlea and their associated nerve cells to enhance temporal variation in the input emphasizing onsets and offsets crucial to detecting consonants. Another portion, which models the nerve cell firing rate related to the characteristic frequency of the cell, emphasizes spectral peaks; this is useful for vowel identification.

The next stage of recognition segments the transformed speech by finding acoustic landmarks in the signal. Regions between the landmarks are grouped into segments on the basis of similarity. A set of phonetic classifiers assigns probabilistic phonetic labels to each of these portions of the speech signal. Different classifiers may be invoked for differentiating among distinct classes of phonemes. Words are represented by a network of phonemes; several networks may be used to encode alternate pronunciations. A pattern matching algorithm similar to dynamic programming matches the phoneme labels of the input against the word networks, deducing word boundaries in the process. Coarticulation rules compensate for some phonemic aspects of connected speech.

SUMMARY

This chapter has detailed the process of recognizing speech by computers. It began with an overview of the recognition process as a set of three basic components: an acoustical representation, a vocabulary or set of templates, and a pattern matching process to measure the similarity of an input utterance with each

of the set of templates. This process was illustrated by considering a simplistic, hypothetical recognizer. The chapter then discussed a full range of possible recognizer configurations differentiated by the specificity of the talker, the style of speaking, and the size of the vocabulary to be recognized. Several techniques important for connected and large vocabulary recognition were then introduced; these included Dynamic Time Warping, Hidden Markov Models, Vector Quantization, and language constraints. The chapter concluded with a descriptive overview of three well-known research projects on large vocabulary speaker independent recognizers.

This chapter was intended to introduce the technology of speech recognition and some of the difficulties it must overcome to be effective. Speech recognition algorithms are under continual development and are useful for an ever-expanding range of applications. Even more important for its deployment, the steady increases in microprocessor speeds enable even the more sophisticated algorithms to be implemented entirely as software thus encouraging the more widespread distribution of applications that make use of recognition.

The next chapter explores the utility of speech recognition, the classes of applications for which it may be well suited, and interaction techniques to make effective use of recognition. Of central concern is the issue of speech recognition errors; error rates remain the primary deterrent to the successful application of speech recognition. But in some specialized application niches speech recognition is already being used to improve personal productivity.

FURTHER READING

O'Shaughnessy covers many topics in speech recognition thoroughly. Furui and Sondhi is a collection of particularly current papers about all aspects of speech processing with particular emphasis on recognition. Although rather technical, it has chapters describing most of the concepts and some of the speech-recognition research projects discussed in this chapter. Waibel and Lee present a collection of previously published research papers about speech recognition. Dixon and Martin is a similar collection of key papers from the 1970s. Another good source of very good working papers are conference proceedings of the DARPA Speech and Natural Language Workshops.

8

Using Speech Recognition

The previous chapter offered an overview of speech recognition techniques and the difficulty of identifying the words in a speech signal. This chapter discusses how speech recognition can be used in applications. It examines the merits as well as the constraints of using voice input, which limit the classes of applications for which it is suitable. Because the dominant difficulty with using speech recognition devices is usually their poor performance, this chapter emphasizes interaction techniques to manage recognition errors. Finally, several case studies in applications of voice input are detailed.

USES OF VOICE INPUT

Voice input can be suitable to an application for one of several reasons. In some circumstances, no other input channel may be available due to the lack of an interface device such as a keyboard or mouse. Even when alternative input devices are available, speech recognition may be more efficient; we can speak faster than we write or type. Voice can also be employed effectively when the user's hands and eyes are occupied with another task.

Sole Input Channel

Regardless of its limitations, voice input is useful when no other input mode is available or when alternative modes present only limited functionality. For this reason, speech recognition currently attracts wide interest from the manually-

disabled population for whom conventional keyboards and mice are extremely demanding.

For many telephone-based voice applications such as those described in Chapter 6, the 12 button keypad is sufficient. Speech recognition can be employed instead of or in addition to touch tones to provide access to a greater number of commands, to make it easier for the caller to remember them, and to enable users to phone in from rotary-dial telephones. Although speech recognition provides a more flexible interface, it is also significantly limited in this context. Because of the limited bandwidth of the telephone audio channel, both high and low frequency sound is lost, removing some cues useful to acoustic classification. Although the quality of telephone networks is improving, some calls, such as from cellular phones, contain significant noise. Even when the telephone circuit itself introduces no noise, users of telephone-based services are often travelers calling from public telephones in noisy locations. Also, as discussed in Chapter 6, because the telephone circuit mixes both the transmitted and received speech it is difficult for the user to interrupt and be correctly recognized.

Speech recognition has been used in a number of telephone-based services usually employing small vocabulary speaker-independent recognition because the identity of the caller is not known. An alternative approach uses speaker-independent recognition or touchtones for the caller to “login” and then loads that caller’s personal vocabulary into the recognizer for the remainder of the call. Some examples of telephone-based recognition are listed here.

- Telephone-based financial services. A caller enters an account number by voice and is told the current trading level of stocks in his or her portfolio or speaks code numbers for a specific stock to hear its price. Because speaker-independent recognition is often limited to the digits zero through nine plus “yes” and “no,” it has not been possible until recently to use a more convenient interface that allows the user to speak stock names.
- Automatic directory assistance for pay-per-use telephone services. Voice menus list the types of services one can call, and the user selects one by speaking choices while traversing the menu. The application then gives a telephone number for access to the chosen service.
- Automated collect call completion. The calling party is prompted to record his or her name. A computer phones the called party, announces the requested collect call, plays the recording of the caller identifying herself, and then asks whether the charges will be accepted. The words “yes” and “no” are recognized; failed recognition results in transfer to a human operator for confirmation.

Speech recognition has enormous potential as the primary means of interacting with new generations of very small hand-held computers. Improvements to laptop computers are mostly in performance; a full keyboard suitable for touch typing places limits on further size reductions. A number of “palm top” computers provide applications such as a calendar and telephone directory but are limited in the amount of data input allowed by their tiny keys. An all-voice

computer, perhaps including a small touch-sensitive display, could be built in a package similar to a miniature tape recorder. At the time this book is being written several manufacturers have introduced families of integrated circuits for digital consumer answering machines; these inexpensive parts are also suitable for a pocket computer.

Auxiliary Input Channel

Speech recognition is useful as an input channel even when a keyboard is available, i.e., when speech is not the only possible means of interacting with a computer.¹

Limited vocabulary speech recognition has been used for a variety of applications in which it essentially substitutes for the keyboard; for a survey of such comparisons see [Martin 1989]. In many traditional screen and keyboard applications such as authoring computer programs, speech recognition has not yet offered a great advantage over keyboard input. However, speech has proven effective in applications in which the user's hands and eyes are otherwise busy, such as baggage sorting, examining printed circuit boards for defects, and entering the results of visual analysis of specimens under a microscope. Speech input is attractive in such situations because of the inconvenience of putting down the object of one's attention for the sake of using the hands on a keyboard.

A particularly strong motivation for voice input is our ability to perform multiple tasks simultaneously if we can employ multiple input modalities. As discussed in some detail in Chapter 6, dividing tasks between verbal and manual channels can result in increased effectiveness and speed by a user. Voice input has particular importance for computer workstations because users are already using a keyboard and often a mouse for a variety of tasks. Computer users are usually occupied manually by typing or using the mouse and visually by the computer display. Although a good typist does not watch his or her hands, most computer tasks involve interacting with data, text, or an operating system for which visual feedback is essential. Because the mouse is used for *relative* positioning, it cannot be operated without attention to the screen. Computer users also often perform multiple tasks in parallel, especially if multiple phases of the same application, such as using a menu and typing data, are considered to be separate tasks.

Computer users, then, can take advantage of voice input for both reasons: being otherwise occupied manually and visually and performing multiple tasks in parallel. In Chapter 6 the same arguments were made to justify potential advantages of voice output. Martin [Martin 1989] demonstrated that for a computer-aided design task involving extensive use of display, keyboard, and mouse, experienced users were able to get more of their tasks done while spending less time looking at the screen when voice input was added. These results motivated

¹The "listening typewriter" is a special case of keyboard replacement which will be discussed in the next section.

Xspeak, a Media Lab interface using speech recognition for navigation in a window system, which is described as a case study later in this chapter.

Another potential use of voice as an auxiliary channel of user input is interacting with a computer at a small distance. A user might wish to start issuing commands such as logging in or initiating a time-consuming task such as sorting electronic mail while walking into the office and hanging up his or her coat. Unfortunately, few recognizers can perform adequately under the highly variable acoustic conditions occurring while the talker moves about in an office, but it may be feasible to electronically “point” an array of microphones at the mobile sound source.

Voice input can also be found in commercial products to speed-dial telephone numbers. The user trains templates and enters the associated digits using the telephone keypad. The user later places a call by picking up the handset and speaking the name to call. Since the caller is about to speak into the telephone during the call, voice-dialing is an intuitive extension of the voice modality. For speech recognition, the telephone handset microphone is well positioned with respect to the mouth; picking it up provides the cue to begin recognizing.

Keyboard Replacement

The previous section argued that there are situations in which speech recognition may be a useful supplement to the keyboard because of other activities a user needs to perform manually. This section considers the potential of speech recognition as a direct keyboard replacement for the purpose of text entry. Although dictation is in many aspects the antithesis of conversational interactive systems, such a “listening typewriter” is the subject of much speech recognition research and the focus of several attempts to develop recognition products.

A listening typewriter may offer several advantages over a keyboard. We can speak significantly faster than we can write or type; we usually speak at several hundred words per minute; an excellent typist can produce 80 or 100 words per minute, and we handwrite at less than half that rate. Of course many of us are not efficient typists, and some professionals are quite adverse to using a keyboard. Even efficient typists may suffer from physical disabilities that interfere with productivity.

At first glance, speech recognition seems to be a superb keyboard replacement. We could imagine such a device useful in any number of large vocabulary computer applications, but we must consider a number of factors in assessing the utility of the listening typewriter. Can speech recognition devices take full advantage of the faster rates of speaking over writing? What portion of the time needed to create a document is consumed by the text input stage? How difficult is it to learn to dictate, and are dictated documents of comparable quality to written ones? Finally, just how extensible is the listening typewriter?

Large vocabulary speech recognition by machine is extremely difficult. As the number of words stored in memory increases, there is an increased likelihood that words will be confused, and pattern matching to select which word was spoken will take correspondingly longer as well. Because a large number of words

may be spoken at any time, word boundary detection is also confounded: As a result many very large vocabulary systems are being developed using isolated word recognition.

Although there are several products and a number of research projects offering some degree of large vocabulary recognition, this work is not yet mature enough to allow an assessment of speech effectiveness for text entry using these devices. The number of similar sounding words confounds recognition in large vocabularies. As a result, current recognizers operate either in less than real time or are constrained to isolated word recognition. But insights into the utility of speech for text input can be gained both from simulations in which a hidden human performs the voice-to-text transcription in real time and by studying how people dictate. Much of the research in this area was done in a series of experiments by John Gould at IBM, who studied dictation as well as simulated listening typewriters.

In one study focusing on the difference between speaking and writing letters, Gould found that speaking required 35% to 75% of the time required for writing and that across both media about two-thirds of the time was spent in planning [Gould 82]. In another study, Gould determined that although people thought their written letters were of better quality, independent judgments did not show this to be true [Gould and Boies 1978]. This work and a similar study [Gould 1978] suggested that dictation is not a difficult skill to acquire.

Gould then provided a simulated listening typewriter to subjects who used it to dictate letters [Gould *et al.* 1983]. He wished to evaluate the importance of vocabulary size as well as the constraints of discrete versus connected speech input. A human transcriber behind the scenes played the role of the speech recognizer and enforced interword pauses for discrete input mode. Depending on the selected vocabulary size, the transcription software determined whether each word should be "recognized." The subject was required to spell an unrecognized word letter by letter. The subjects included both expert as well as naive dictators.

Gould measured the composition time and evaluated the quality of the letters produced by his subjects who used the simulated recognizer. He found that by both criteria speaking letters was comparable to hand writing them. The fact that expert subjects (experienced with dictation) were not significantly faster than novices suggests that the experimental setup was not as efficient as simple dictation; using any of the simulated technologies, someone talking to a computer was slower than speaking into a tape recorder.

Although Gould's subjects preferred connected over discrete speech, the difference was not as large as one might expect given the requirement to pause between each word. This may be due in part to the nature of the experiment. In one task, subjects were required to write a final draft version of a letter. If the recognizer made an error, the user's only recourse was to back up and delete each intervening word, replace the misrecognized word and then repeat all the deleted words. This may have encouraged subjects to pause between words until each recognition result was displayed on the screen, even while using the connected speech mode. This may also partially explain why expert dictators did not perform as well as might have been expected, given their experience.

What do Gould's studies indicate as to the suitability of speech as a keyboard replacement? It seems easy to learn to input text by voice, and the resulting documents are of equal quality to those written or typed. Although isolated word recognition is significantly slower than connected recognition, it does appear to have utility and may be acceptable in an office environment. But there is more to writing a passage than entering words. Although speaking may be five times faster than handwriting and two to three times faster than an experienced typist, *composing* a letter takes a significant portion of the time spent creating a document, thus diminishing some of the speed improvement afforded by voice input. After initial authoring, documents may require editing, which Gould noted is easier with handwriting. Some editing functions such as moving a cursor to select a region of text may be awkward to perform by voice, and it is a challenge for the recognizer to detect when the user wants to switch between text input and editing.² Finally, it must be noted that although Gould's studies compared voice to handwriting, typing is becoming an increasingly common skill and is much faster than handwriting (although still much slower than fluent speech).

Despite these objections, the listening typewriter would still undoubtedly be useful for some users and in some work environments. Research in this direction hastens the arrival of large vocabulary general purpose recognizers as well. However, specific assumptions about the structure of written language used to constrain the recognition task may render the recognizer less useful for other potential applications. For example, the language, or sequences of words, that a user would speak to a computer operating system would possess its own unique syntax very different from the business correspondence used to seed the trigram probability tables. In addition, operations such as selecting a file name or entering the name of a new file to be created much less constrain the choice of the next word than natural languages and therefore are harder to recognize. The terseness of text-based computer interfaces is efficient for the skilled user, but by removing redundancy they comprise a more difficult language for recognition.

This section has considered the production of text documents by voice in contrast to keyboard or handwritten input. Voice as a document type was considered in depth in Chapter 4. Although dictation and using a speech recognizer instead of a keyboard are similar tasks, both involve quite different authoring styles than recording short voice memos that are never converted to text. Comparisons of creation time with quality across voice and text documents may not be very meaningful as the document types are most likely to be used for rather different styles of messages.

²To achieve acceptable rates of recognition with such large vocabularies, recognizers must take advantage of the context in which the word occurs. The requirements imposed by assumptions of context, both preceding and following the word just spoken, interfere with providing immediate editing commands. For example, the Tangora recognizer described in the previous chapter does not report a word as recognized until at least one additional word has been spoken to afford adequate context for effective recognition.

SPEECH RECOGNITION ERRORS

The primary limitation of speech recognition is the accuracy of current recognizers. As a result, coping with errors dominates user interaction techniques. The difficulty of recognizing speech further limits the range of applications for which it is suitable. Accuracy is so impaired by noise that speech recognition can be used only with great difficulty in even moderately noisy environments or over impaired communication channels such as cellular telephone connections. Speech recognition applications must operate with smaller vocabularies, making it difficult to support broad functionality, in order to attain acceptable recognition rates. Even the most simple application must deal with recognition errors; for complex applications error detection and correction may require more programming than the portion of the application that actually performs the user's suggested action.

Classes of Recognition Errors

Speech recognition errors may be grouped into three classes.

1. *Rejection*: The talker speaks a word from the recognizer's vocabulary but it is not recognized.
2. *Substitution*: A word spoken from the vocabulary is recognized as some other word.
3. *Insertion*: Extraneous words are recognized in response to stimuli such as the talker's breathing or lip smacking or perhaps environmental noise such as a door slam or keyboard clicks.

Different strategies are required for coping with each type of error. All errors are more likely to occur when using connected recognition, large vocabularies, and speaker-independent devices. Because of rejection and insertion errors, a connected recognizer may even report incorrectly the *number* of words spoken in addition to making a substitution error on any particular word.

Users want the best recognizer they can buy; however, it is very difficult to measure error rates of a particular device. Error rates will be a function of the background noise level, the vocabulary set, the speaker's cooperation, attitude, and experience. Most manufacturers claim recognition accuracy in the high 90% range, but these rates are unrealistic and rarely realized in practice. Accuracy measurements across recognition devices are a meaningless basis for comparison unless the same test conditions are reproduced for each measurement. This requires using a standardized test, with both test and training data taken from the same audio source, e.g., a test tape or CD.

Real users rarely achieve the rates claimed for a device by the vendor. Reducing accuracy to a single metric can be misguided because rejection, substitution, and insertion errors may have different impacts on the underlying application. The difference between 95% accurate recognition and 98% may be very significant in terms of the difficulty in improving the recognition algorithm (to halve the error rate), but this reduction is much less significant for user interface design; in both cases, the recognizer is correct most of the time but makes occa-

sional errors. The speech input interface still has to cope with errors or it will have little utility.

Factors Influencing the Error Rate

Recognition errors are caused by many factors; some are more obvious than others. Techniques can be employed to minimize errors, so it is important to recognize early in a design what factors are likely to contribute to an increased error rate. Understanding the causes of errors also helps us appreciate why some applications are more amenable to speech input than others. The necessity of detecting and correcting errors precludes the use of recognition from some applications where it may seem ideal at first glance.

Word Similarity

Words which sound similar, such as “bat” and “bad,” are difficult for humans to distinguish, so it should not be surprising that they are also frequently confused by a speech recognizer. Without a thorough understanding of the acoustic representation used by the recognizer, it is difficult to know which words sound similar, but a good first approximation is that words containing similar phoneme sequences are likely to be a source of recognition errors.

As previously mentioned, as the number of words to be recognized increases it becomes increasingly difficult to distinguish between them. Longer words contain more phonemes and hence are more likely to contain cues allowing them to be distinguished from other words. For example, the words “impossible” and “improbable” contain nearly identical sequences of phonemes, but the fricative “s” is a robust cue for differentiation. Errors can be minimized by careful selection of the vocabulary set to be used in an application, avoiding short and similar sounding words. When using a discrete word recognizer, it is helpful to string together short words such as “to the right of” into phrases which are trained and spoken as if they were single words.

But modifying one’s vocabulary to optimize recognition performance may not be possible or may detract from the advantages of speech input. “Five” and “nine” could be confusing words for recognition because of their common vowel, but if the application requires the user to enter a telephone or credit card number, the use of these words cannot be avoided. Although “affirmative” and “negative” are more easily distinguished than “yes” and “no,” they take longer to pronounce and more importantly are not part of most of our daily vocabularies. In situations in which speech recognition is essential, users may be amenable to using specialized vocabularies; however, this is at cross purposes to the claims of naturalness and ease of use for voice input.

Acoustic Environment

Noise interferes with recognition—although some recognizers are more sensitive to noise than others. Environmental noise varies widely; there are vast differ-

ences in the amplitude and spectral characteristics of the acoustical ambiance in an office, on a factory floor, or inside an automobile. In a noisy environment, special noise-canceling microphones may assist recognition (see Figure 8.1). These microphones attempt to remove background noise from the speech signal and are usually worn on the head or held near the mouth. This position is a hindrance in many circumstances.

Impairments in the acoustic channel may be constant or vary with time; constant characteristics are easier to model and hence are easier to compensate for. Speech recognition over the telephone provides an example of each. As discussed in Chapter 10, a telephone circuit transmits only the portion of the speech spectrum below approximately 3500 Hertz,³ so a recognizer designed for telephone use must not rely on acoustic cues based on frequencies above this. This limited bandwidth is a predictable characteristic of the telephone channel. In addition, if a call is placed from a public telephone, there may be highly variable background noise such as traffic sounds, flight announcements, or the caller at an adjacent telephone. Because it is difficult to model such noise, it is also difficult to compensate for it.

³A manifestation of limited bandwidth is the increased difficulty we have in identifying a person by voice over the telephone.

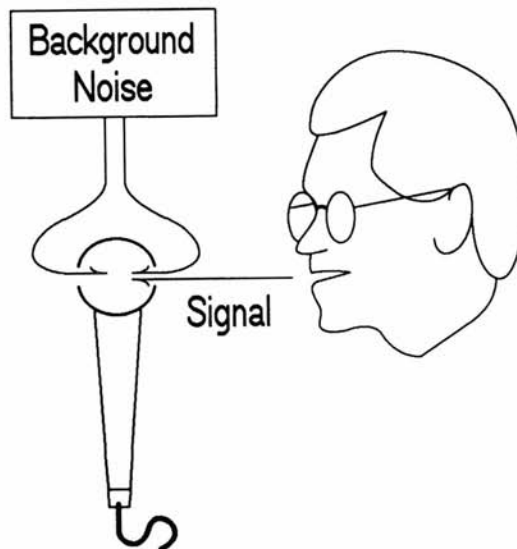


Figure 8.1. A noise-canceling microphone. Background noise enters the microphone in both the front and rear openings, but the user's speech is detected primarily at the nearer opening. This allows the microphone to subtract the background noise from the combined input of speech and background noise signal.

Talkers' Characteristics

Recognition simply works better for some people than others; this is what is sometimes referred to as the “sheep-and-goat” problem. Some users, the “sheep,” can achieve high recognition accuracy. They are more cooperative and adaptive with their speech habits or perhaps simply more consistent in how they speak. Others, the “goats,” have considerable difficulty with recognition. They may experience “microphone fright” or be resentful of the implications of a computer assisting them on the job. Many people compensate for errors during recognition by speaking louder, which may cause distortion by overloading the audio circuits, or by trying to speak more distinctly, which may make their speech less similar to the templates that they trained while relaxed. A user with a cold and congested nasal passages may need to train a new set of templates to achieve acceptable recognition due to the changed spectrum of the vocal tract transfer function.

INTERACTION TECHNIQUES

It is impossible to build an effective speech recognition application without taking care to minimize errors if possible and incorporating a means of coping with recognition errors. The previous section catalogued factors influencing the error rate; some of these may be at least somewhat under control of the application, such as utilizing a recognition vocabulary that avoids words that sound similar. This section also discusses other activity that can be incorporated into the user interface to diminish error probabilities. But errors will happen. This section emphasizes interaction techniques that can be employed by the user interface to allow applications to utilize recognition despite the errors.

Because recognition errors may be due to different causes (insertion, rejection, and substitution errors were just described) multiple error recovery techniques may be needed in a single application. There are two aspects of coping with errors. **Error detection** identifies that an error has occurred. Detection can be triggered by the user's apparently requesting an improper or meaningless action by the system or by the recognizer's reporting that a rejection error has occurred. **Error correction** techniques can then be employed to rectify the error and allow the application to determine what the user actually requested. Recognition errors may not be readily detectable by an application. If the user says “Call Barry” but the recognizer reports “Call Harry,” this semantically correct substitution error might easily pass unnoticed, so the application must provide for user-initiated error correction as well.

This section focuses on techniques, largely conversational, for understanding the user's spoken input. If an error is detected, what should a user interface do about it? If the user detects the error, what corrective action does the application provide? This section discusses techniques to minimize recognition errors based largely on the observations made above on factors contributing to errors. It then examines confirmation strategies whereby the user is informed of what the application is about to do and has an opportunity to veto any command triggered by

erroneous recognition. The final part of this section presents methods whereby either the application or the user may invoke iterative techniques to detect and correct recognition errors from input command sequences already in progress.

Minimizing Errors

Recognition errors result in impaired application performance and increased user frustration. Whenever a recognition error occurs, a repair strategy must be invoked by either the user or the application. Repair inevitably necessitates the user speaking more words, which may in turn be misrecognized in addition to the error that the dialogue is meant to correct. As the error rate increases, it takes correspondingly longer to complete whatever task is supported by the application, and user satisfaction with speech recognition plummets. Error rates can be diminished by compensating for the factors resulting in errors; several factors were discussed in the previous section:

- Careful selection of the recognition vocabulary.
- Control of environmental acoustics and the use of a head-mounted noise-canceling microphone.
- Training users to speak clearly and consistently.

Several additional techniques can be used to further minimize recognition errors.

Vocabulary Subsetting

Many recognizers allow the application to specify that only a subset of the full vocabulary should be available for recognition at a particular moment. This dynamic modification of vocabulary size can result in significant performance improvements if the structure of the task permits. Many speech recognition applications use speech input much like filling out a multiple choice form; at any juncture in the user interaction there are a small number of choices of what to say next. For example, if the user is asked to speak a telephone number, recognition can be restricted to the digits plus a few key words, such as “area code” and “quit.” If the application asks a yes-or-no question, the active vocabulary can be reduced to just the two possible responses.

Language Model Subsetting

Most current large vocabulary speech recognizers employ language models to improve accuracy by specifying all possible input sentences prior to recognition. This is similar in concept to vocabulary subsetting where the subsetting is done on a word-by-word basis while attempting to recognize a sequence of words. The application can facilitate this process by activating and deactivating portions of the language model (sometimes called “contexts”) during the course of interaction. For example, at the stage of a transaction where the user is expected to

Speak a dollar amount, the recognizer can be told to use only the small portion of its language model dealing with dollars, cents, digits, etc.

Explicit Indication of Attention

In many acoustic environments background noise is highly variable, which results in spurious insertion errors when short, loud sounds are picked up by the microphone. Insertion errors also occur if the user speaks with another person such as a telephone caller or a visitor. Such errors can be minimized by disabling recognition when it is not in active use.

One simple means of disabling recognition is the addition of a mechanical switch on the microphone cable. A hand-held device employing speech recognition may require a push-to-talk button in a noisy environment. If the user is at a computer, a mouse button may be used the same way. But any mechanical switch detracts from the benefit of being able to use recognition while one's hands are otherwise occupied.

Another alternative is to use voice commands such as "stop listening" and "pay attention." After the user says the first of these commands, recognition results are discarded until the second is spoken. Note that it is important to use phrases for this software switch that are unlikely to be spuriously recognized themselves. Another disadvantage of the stop listening approach is that the user is likely to forget that recognition is disabled until speaking several times with no apparent results.

Some versions of CMU's Sphinx recognizer used a key word to indicate attention and trigger recognition: "Computer, when is my meeting?" "Computer, cancel the meeting." Recognition control can also be implemented using the keyboard or mouse if the user works at a computer console. Finally, it is possible to use the direction in which the user speaks as the focus mechanism. The Media Lab's Conversational Desktop project [Schmandt *et al.* 1985] used such an approach; the user turned toward the screen when he or she wished to be heard, and an array of microphones detected this change of attention.

Confirmation Strategies

The need to cope with the high likelihood of errors must play a lead role in the choice of interaction techniques. The interface designer must consider the consequences of the application performing an incorrect action on the basis of misunderstood input and how to improve the chance that the user will notice the error before doing further damage. By setting a high rejection threshold, an interface designer can decrease the number of misrecognized words but at the cost of an increase in rejection errors. The user can be more confident that the application will not err, but it will likely take longer to perform the correct action.

Whether it is faster to reject many possibly correct utterances or accept errors and allow the user to correct them depends on the nature of the task, the complexity of the input, and the time required for recognition. Depending on the prob-

abilities of making errors with particular vocabulary sets, different correction or confirmation techniques can be modeled as Markov processes and their behavior predicted [Rudnicky and Hauptmann 1991]. But for some applications an error cannot be repaired. This is the case when the computer is going to perform some action with dangerous or irrevocable side effects (launching a missile or deleting a file). Such performance errors can be avoided by offering the user a chance to cancel a misrecognized request before the application has acted on it. At the cost of requiring greater participation from the user, several different confirmation strategies can be invoked to avoid making a mistake.

Explicit Confirmation

When misrecognition would cause an irrevocable performance error, explicit confirmation may be required from the user, elicited by echoing the request as recognized: "Please confirm that you wish to delete the file named 'book'." If the user's response is then limited to "yes" or "no," vocabulary subsetting can be employed to decrease the chance of misrecognition. In addition, the rejection threshold can be increased for the reply for added confidence that the user really wishes to proceed with a dangerous action.

Explicit confirmation is slow and awkward, requiring responses and user feedback with each interaction. As described here, it offers the user little recourse for misrecognition except the opportunity to try again without causing damage. Hence, it should be used sparingly. If the consequences of system action and recognizer accuracy together suggest use of explicit confirmation on every user request, the task is likely to be ill suited to speech recognition.

A more flexible variation is to use explicit confirmation only when the user's choice is implausible or the recognition score for a word is low. In such circumstances an effective technique is to ask "Did you say . . . ?" and temporarily switch to explicit confirmation mode.

Implicit Confirmation

An alternative to explicit confirmation is *implicit confirmation*, in which the application tells the user what it is about to do, pauses, and then proceeds to perform the requested action. The application listens during the pause, thus giving the user the opportunity to cancel the request before any action occurs. If the request takes significant time to perform, such as dialing a number from a cellular phone, the application may be able to begin performing the task during the pause; if the user cancels, the call attempt is abandoned before the call has had time to go through.

Implicit confirmation allows the system to perform more effectively as recognition accuracy improves. The user is required to intervene only in the case of an error; otherwise the request and resulting action proceed smoothly. Implicit confirmation could be slightly slower than explicit confirmation as the application must pause long enough to provide ample time for the user to respond. The user can be allowed to terminate the pause by saying "OK"; this is similar to the

optional termination techniques discussed in Chapter 6. If the user always responds, this is identical to explicit termination. Although the application is idle during this pause, the user can be performing other tasks and thinking about what to do next. Because the application has confirmed what it is about to do, the user need not devote additional attention to the task.

Error Correction

The previous section described confirmation techniques to ensure that erroneous recognition did not occur or was canceled by the user. This was based on a binary judgment by the user as to whether recognition was correct; rejection caused the entire input to be discarded. While this may be necessary in error-critical situations, a more effective strategy is to attempt to detect recognition errors and invoke repair dialogue to resolve misrecognized words.

An error may be noticed by the application or it may escape detection only to be noticed by the user. The application may detect that an error occurred because the resulting sentence is syntactically ill formed or semantically meaningless; techniques for such analysis are discussed in Chapter 9. Or it may be that an apparently well-formed request is meaningless in the current state of the application and its associated data. For example, consider a spreadsheet in which the rows are numbered and the columns are lettered. “Add row five and row B” is a syntactically well-formed sentence in and of itself but does not describe an operation that can be performed on this spreadsheet. “Add row five and row twelve” could be a valid operation, but if row twelve has not yet had values assigned to it, the operation still cannot be performed. An application supports a more limited set of operations than can be described with the associated vocabulary, and the current state of the application further constrains acceptable user input.

Depending on the class of recognition error (rejection, insertion, or substitution), the application may be faced with an incomplete utterance, a meaningless utterance, or a valid request that is unfortunately not what the user asked. If the application detects the error, how can it determine what the user wants? If the user detects the error, how can the application allow the user to make repairs?

Additional Recognition Information

In addition to identifying which word was detected, some recognizers supply additional information that can be used during error recovery. One such piece of useful information is the word that is the **next best match** to the spoken input. The similarity scores for each word indicate how reliable that choice is and how plausible the second choice is if the first is deemed to be in error. If the first choice is not meaningful in a particular context, the second choice may prove to be an acceptable alternative.

Some recognizers are capable of reporting the **confusability** of the words in its vocabulary. Confusability is a measure of how similar each word is to every other word in the vocabulary and gives an indication of the probability that any other

word will be substituted. If the recognized word is inappropriate in the current context but another word that is highly similar is appropriate, the application could use the alternate instead of the recognized word.

In the spreadsheet example, the phrase "row B" was suspect. Either second choice or confusability information could be used to indicate whether the user actually spoke "row three" or "column B," for example.

Alternate Input Modalities

In addition to speech, other input channels may be used to point or select the data represented by an application. Hauptman found that users were able to make effective use of pointing in combination with limited vocabulary speech recognizers to describe graphical manipulations [Hauptmann 1989]. Pointing devices may be two dimensional (mouse, stylus, touch screen, joystick) or three dimensional (gesture input using a variety of magnetic or ultrasonic techniques). Eye trackers report the angle of the user's gaze and can be used to manipulate an object by looking at it [Jacob 1991].

The selection gesture may be redundant to some words in a spoken command; this is especially true of deictic pronouns ("this," "that," "these," "those"). If the gesture can be correctly synchronized with the reporting of speech recognition results, the gestures can be used in place of any pronouns not recognized. This technique was first used in Put That There, a project described later in this chapter. Carnegie-Mellon University's Office Manager (OM) system used mouse-based pointing as a means of explicit error recovery [Rudnicky *et al.* 1991]; after recognition, a text display of the user's request was presented, and the user could click with the mouse on a word to be changed. Voice has been combined with gesture in a three-dimensional model building task [Weimer and Ganapathy 1992, Weimer and Ganapathy 1989]. Starker and Bolt combined speech *synthesis* and eye tracking into a system that discussed what a user was looking at [Starker and Bolt 1990]; and more recently the same research group has incorporated speech recognition as well [Koons *et al.* 1993].

Dialog With the User

The errors that an application detects result in a meaningless or incomplete request. The application must then employ techniques to elicit additional information from the user. The simplest but not very effective technique is to ask the user to repeat the request. If the user repeats the request word for word, it is quite possible that the same recognition error will occur again. If the user instead rephrases the request, different recognition errors may occur, and it is challenging to merge the two requests to decide what the user really wanted.

It is better for the application to ask the user one or more specific questions to resolve any input ambiguity. The phrasing of the questions is critical as the form in which a question is asked strongly influences the manner in which it is answered. The goal of efficiency suggests that the questions be brief and very direct. The goal of minimizing user frustration would also suggest asking ques-

tions that can easily be answered by a single word because words spoken in isolation are more easily recognized than connected speech. Consider a schedule application as an example. The user requests "Schedule a meeting with Rebecca tomorrow at three o'clock"; however, the word "three" is rejected. Because the application requires that a time be specified to complete a schedule request, the error can be detected. In return, the application could ask "At what time?"

Echoing

Asking a terse question is fast but conveys little information about the words that the application assumes to be correct. In the above example, if the application responded "What city?" the user would realize the application had probably completely misunderstood. But if instead of "Rebecca" the speech recognizer had detected "Erica," the query "At what time?" could not convey the occurrence of this error to the user.

Echoing is a technique to offer extensive feedback to the user. For example, the application might have asked "At what time do you wish to meet with Erica tomorrow?" so an alert user could notice the error. Even when the application receives an apparently correct command sequence, echoing it to the user provides an opportunity for the user to take further action. The major drawback of echoing is that it takes time and makes the application much more verbose.

Backtracking

When recognition errors result in a well-formed and plausible command, an application cannot detect the error and will perform the wrong action in the absence of a confirmation strategy. The user may recover from such an error by backtracking, in which the user simply cancels a request or asks the application to "undo" the action. This is appropriate only when the command can, in fact, be canceled with little or no penalty. To implement backtracking, the application must remember its state before each round of user input; the complexity of this memory is task dependent. Backtracking is a corrective behavior invoked entirely by the user but assisted by the application's ability to track its state.

CASE STUDIES

What do these factors lead us to conclude as to the suitability of speech recognition to real world work situations? Speech is fast and can be effective for text entry provided the recognizer performs adequately. Performance is enhanced by constraining the recognition choices at each word boundary, i.e., by minimizing the perplexity of the recognition task. This implies that recognition for free-form dictation is more difficult than for application-specific tasks.

Recognition is often suitable for tasks in which one's hands and eyes are busy, providing a valuable auxiliary channel for tasks such as computer-aided design. But note that recognition in this context is more suited to selection of one of a

small number of items from a list (e.g., a menu) or selecting an object from some number on display than for continuous tasks such as positioning a cursor.

Recognition is particularly difficult in noisy situations, such as in an automobile or on a factory floor; under these circumstances users may be forced to wear head-mounted microphones. Recognition may be possible in a relatively quiet office environment, but privacy concerns and social conventions make it unlikely we will talk to our computers if we share our office.

This chapter has discussed both the characteristics of applications that may make them successful candidates for speech recognition as well as interactive techniques required to allow users to cope with recognition errors. This section considers two case studies to illustrate these themes. The first case study examines Xspeak, which used rudimentary isolated word recognition to manage windows on a workstation display. Xspeak illustrated how commercially-available recognition may already perform adequately for deployment into normal office environments⁴ and discusses experiences of real users incorporating recognition into their jobs.

The second case study, Put That There, demonstrates an early application of multiple error management techniques. Put That There employed connected speech recognition, and the nature of the application required very different error management than Xspeak. Put That There was one of the first speech user interfaces to merge both recognition and synthesis into a conversational system.

Xspeak: Window Management by Voice

This chapter has contrasted the large vocabulary listening typewriter with smaller vocabulary application-specific recognition. But few workstation-based applications use recognition, and the discussion of error handling techniques should make it clear that incorporating recognition into an application requires considerably more programming than simply reading from an additional device for input in addition to the mouse and keyboard. Furthermore, most desktop computers are now running window systems, allowing users to employ multiple applications during the same work session. If several applications use speech recognition, will they compete for the microphone? Recognition results must be distributed to the correct application process using some indication of the user's intention, such as knowledge about the vocabulary employed by each application, or which window is currently active.

Window systems are ubiquitous on computer workstations. The window system divides the screen into multiple regions ("windows"), each of which can be devoted to a single application. An operating system may allow multiple programs to execute simultaneously; the window system allows each of them to display output without disturbing the others. A typical workstation user is likely to run many

⁴Since this project was completed, a number of products to control windows by voice have appeared on the market.

applications as a matter of course, each utilizing one or more windows (see Figure 8.2). These might include a clock, calendar, mail reader, text editor, and perhaps a debugger with a graphical user interface.

The window system may allow windows to overlap or may tile them such that all are visible simultaneously. Overlapped windows are currently more popular; the added load on the user to keep track of partially “buried” windows allows more applications to make effective use of the screen simultaneously [Bly and Rosenberg 1986]. A **window manager** controls which window receives keyboard input or **focus**. A **real-estate-driven** window manager assigns input to the window on which the mouse cursor appears; **click to focus** requires the user to click a mouse button on a window to shift the focus there. The mouse is also used to hide and expose windows and to specify their size and location. A third use of the mouse is user input for applications supporting direct manipulation interfaces with buttons, scroll bars, sliders, and toggle switches.

Xspeak was a Media Lab project that explored an alternative use for voice in the multiple-application workstation environment [Schmandt, Ackerman, and Hindus 1990]. Instead of focusing on the applications themselves and rewriting them to support speech recognition, Xspeak instead used voice input to supplement the mouse for switching *among* applications under X windows. Xspeak provided an extension to window systems by matching limited recognition capabilities to a well-defined set of window operations. Xspeak is an example of an application for which small vocabulary speaker-dependent isolated word recognition is adequate. Although Xspeak was implemented using external recognition

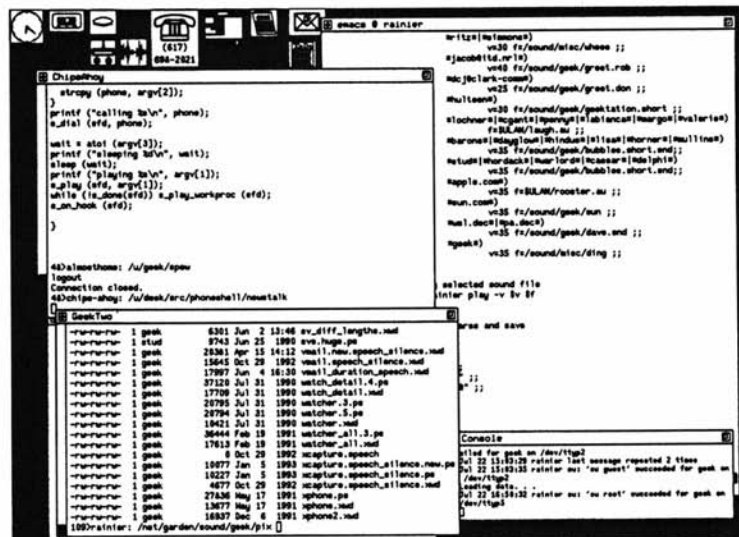


Figure 8.2. Many windows may be displayed simultaneously on a single screen.

hardware, suitable software-based recognition requiring no additional hardware is already becoming commercially available.⁵

Xspeak Functionality

Xspeak provided for the use of voice for the first two mouse functions previously mentioned: focus management and navigation among windows that may be obscured by other windows. Xspeak worked with real-estate-driven window managers supporting overlapped windows. Each window had a name; speaking its name exposed the window. Xspeak then moved the cursor into the window so the window manager assigned it the keyboard focus. Thus the user could move among applications sending keystrokes to each in turn without removing his or her hands from the keyboard. Additional commands allowed the user to raise or lower the window in which the cursor currently appeared.

Xspeak employed a small window to provide both textual feedback to the user and recognition control capabilities (see Figure 8.3). As each word was recognized, it was displayed in this window. In ordinary use this window was ignored as the window reconfiguration provided adequate feedback, but the display panel was useful when recognition results were particularly poor and the user needed to confirm operation of the recognizer. Through the control panel the user could name a new window; this involved clicking on the window and speaking a name that was then trained into a recognition template. The control panel also provided a software “switch” to temporarily disable recognition when the user wished to speak with a visitor or on the telephone. Another button brought up a configuration window, which allowed the user to retrain the recognizer’s vocabulary, configure the audio gain of the speech recognizer, and transfer the recognition templates to disk.

Xspeak was an application not a window manager (see Figure 8.4). It worked by capturing recognition results and sending X protocol requests to the server. As with any client application, these requests could be redirected through the window manager, and the window manager received notification of the changes in window configuration or cursor location. Because Xspeak did not replace the window manager, it could be used across a wide range of workstation window system environments.

Why Speech?

Xspeak was motivated by the belief that window system navigation is a task suitable to voice and well matched to the capabilities of current speech input devices. Using speech recognition to replace the workstation keyboard would be very difficult; it would require large vocabulary speech recognition devices capable of dealing with the syntaxes of both English text as well as program text and oper-

⁵Xspeak has recently been revived using all-software recognition. This greatly facilitates deployment of the application.



Figure 8.3. Xspeak control panel.

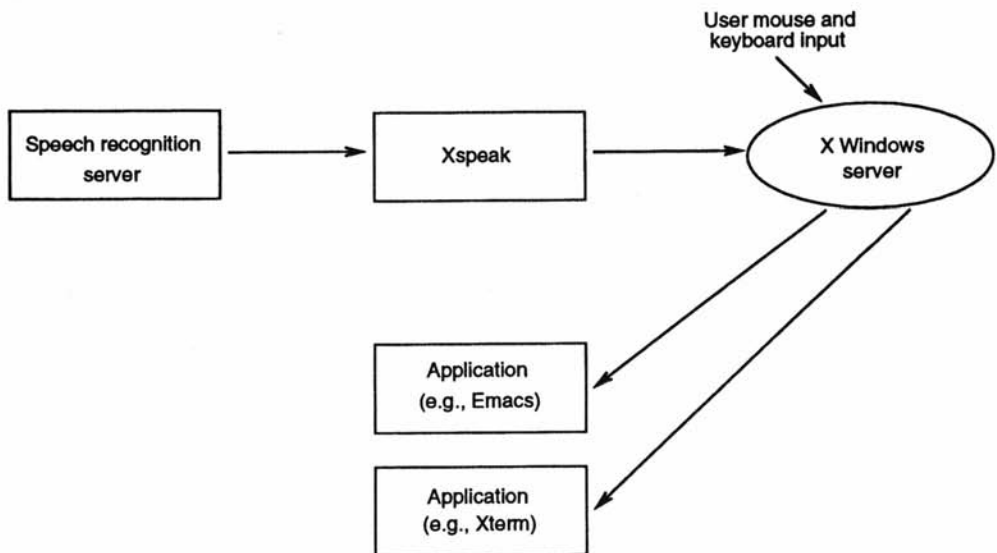


Figure 8.4. Xspeak cooperated with other processes under X windows.

ating system commands. In contrast, a typical user employs a relatively small number of windows, and because the user need speak only the name of the window, isolated word speech recognition suffices for the task. Using speech recognition to replace the mouse takes advantage of the fact that the user's hands and eyes are busy on the keyboard and screen. Using voice input saves the trouble of finding the mouse manually, performing an action, and then repositioning the hand on the keyboard.

Voice input also overcomes the dimension mismatch between windows and the mouse. The mouse is a two-dimensional input device, but windows are two-and-a-half dimensional; windows are planar but the plane of any window can be in front of or behind the planes of other windows. Once a window gets buried, the mouse must be used to move other windows out of the way until some part of the buried window is exposed, so mouse actions increase with the number of windows. But speech input bypasses the dimension problem and exposing windows is no more (or less) difficult than recalling an element from a list of names.

Finally, it was hypothesized that the divided attention theories previously mentioned would support improved performance by allocating multiple input channels to the task of working under a window system. To the extent that navigation and focus management are distinct tasks from the operations being performed inside the selected windows, dividing these tasks across modalities should enhance user performance.

Xspeak in Use

A small number of student programmers used Xspeak during their normal workday for several months and were observed by interviewing, videotaping, and logging of window system events [Schmandt *et al.* 1990]. These observations support the hypotheses about the utility of voice in this task only weakly, yet clarify some of the difficulties of speech recognition.

The dominant factor in user interactions and acceptance of Xspeak was errors. Even in this simple task, recognition rates were abysmal (often less than 80% correct), in large part due to the microphones selected. Because it seemed doubtful that programmers (or the real users toward whom an interface such as Xspeak would eventually be targeted) would wear head-mounted microphones, high quality directional microphones were positioned on stands next to the computer monitors. These microphones admitted much nonspeech noise, and the recognizer responded with a propensity for insertion errors due to sounds such as the keyboard, doors closing, and conversations in the hall. An insertion error would result in the user's keystrokes going into the wrong window; unless noticed immediately this could cause serious damage. To eliminate insertion errors, rejection thresholds were set rather high, resulting in the tendency towards rejection errors reflected in the recorded error rates.

Despite these errors, several users expressed a preference for the recognition-equipped workstations. Those realizing the most advantages from Xspeak were the very active users who kept themselves busy with tasks running in many windows. One such user had already developed techniques using an "icon manager"⁶ to control windows and found no added value to speech input. One user employed very few windows and spent much of his time at the workstation during this study thinking and never made much use of recognition. All users abandoned the

⁶An icon manager displays a control panel listing the names of all windows. The user can make windows appear and disappear by clicking on this icon manager menu.

microphone-equipped consoles when significantly faster workstations became available elsewhere in the laboratory. Nonetheless, a majority of the users were able to use Xspeak effectively and incorporated it into their daily work routines. Some were extremely enthusiastic and competed for access to Xspeak.

Users did make frequent use of the control panel to test the recognizer and retrain words when recognition was especially poor. Users needed help selecting a vocabulary set appropriate for recognition; one user tried to train nonspeech sounds such as a sigh and a hiss as window names with no success. Summarizing the observed usage and opinions garnered from interviews, Xspeak was certainly intriguing and was perceived positively despite the high error rates, but it simply did not work reliably enough to change work habits. Most users thought that Xspeak was much faster than using a mouse (when recognition succeeded), despite measurements showing that for the most common very simple window actions (switch focus between two exposed windows) the mouse was actually somewhat faster.

The last observation might lend some credence to the divided attention theory, as performance would have to be measured by software production quantity or quality and the perceived speed of recognition might indicate that it interfered less with the programmers' cognitive processes. But the users' strongest complaint (after poor recognition accuracy) was that they wanted to use speech to interact with applications, not just select between them. The clean division of mouse functionality presented above did not dominate the users' perceptions, which suggests that using recognition for window management is just one aspect of multimodal input at the workstation. Although an interaction language to integrate speech recognition into application commands by simulating X input events and distributing them to application windows was designed, it was not implemented.

Put That There

An early example of an application employing several of the techniques described in this chapter was Put That There, done at an MIT Media Lab predecessor, the Architecture Machine Group, in 1980 [Schmandt and Hulteen 1982, Bolt 1980]. Put That There was an exploration into the use of multimodal input and conversational techniques to resolve recognition errors. While using this application, the user sat in front of a large-screen video projector and used voice and gesture to manipulate ships on a display of a map. Gesture was tracked via a magnetic six-degree-of-freedom sensor worn on the user's hand. The application responded with visual as well as aural feedback using synthesized or digitized speech.

Put That There was a command-and-control application that, depending on the selected database, let the user either create, move, modify, name, or destroy a fleet of ships in the Carribean or build towns, forts, and churches around colonial Boston. The work described here was based on an earlier prototype allowing for manipulation of abstract shapes on a blank screen. Although the prototype presented an impressive demonstration of the technology, the lack of voice response made for an interface only marginally usable: If a single word of an utterance was

not recognized, the mute application offered no feedback, and the user's only recourse was to repeat the utterance until something was recognized.

Speech was the primary input to Put That There, via a connected-speech recognizer with a vocabulary size of about a hundred words. The version described here accepted such utterances as these.

- Create a large red sailboat north of Haiti.
- Delete the cruise ship.
- Move the yellow oil tanker east of the Bahamas.
- Copy that (pointing to a ship) . . . north of that (pointing to another ship or a feature on the map).
- Where is Havana?
- Move the yellow oil tanker south of the large green sailboat.

The application's main input loop waited for recognition results while tracking the hand and recording gestures as they were detected. When speech was recognized, the first stage of the application's parser analyzed the input word by word. Each word was classified as an instance of a small class of semantic types from a grammar very specific to this application. Examples of these classes include the following.

- *Command*: the requested action.
- *Movable-object*: a type of ship, e.g. freighter, oil tanker.
- *Color*.
- *Size*.
- *Destination*: either a map location or another ship.
- *Destination-relative*: prepositional phrases such as "north of" or "below."

The parser generated a frame, i.e., a data structure describing a request in progress, which contained slots for the various semantic classes filled in with the specific instances of the words heard. The input loop then dispatched to command-specific procedures for semantic analysis. These procedures were hard-coded program segments that analyzed the frame to determine whether it adequately specified the requested action; for example, a "move" command requires an object to be acted upon as well as a destination. At this point, pronouns such as "that" or "there" were associated with screen coordinates by mapping gestures to objects identified by the bounding boxes of the various targets on the screen. Even if the pronoun were missing due to a rejection error, a recent gesture could satisfy the target specification requirement if it uniquely identified an object or position.

Database routines were invoked to associate descriptions with objects. If the user spoke "the red freighter" and this uniquely matched a ship, parsing could continue. If no ships matched the description, Put That There would ask "Which object?" In the case of multiple matches it would ask "Which one?" These terse responses were barely adequate for guiding the user to an unambiguous request; they would have benefitted from echoing or other explanatory behavior to help understand the nature of the misrecognition.

If an utterance was complete, then these command-specific analysis routines invoked action routines to update the ship database and modify the display accordingly. Most often, however, one or more words would have been misrecognized, resulting in an apparently incomplete request. Each semantic analysis routine included code to generate a query to the user; these queries were designed to elicit single-word responses because the recognizer exhibited much better performance on isolated word input. The queries such as "Where?", "Which one?", and "Relative to what?" were spoken to the user by a primitive speech synthesizer. Since there was a limited set of queries, the synthesizer was later replaced with higher quality digitized speech stored on magnetic disk.

Parsing and semantic analysis were **re-entrant**; after a query was spoken to the user, additional input was gathered from the main input loop and inserted into the utterance frame exactly as before. Since the frame was not initialized after the query, any new information was merged with the old and the semantic analysis routines could then examine the frame again. This query-response-parse cycle continued until an action could be executed or the user reinitialized the process by saying "restart." The net effect was that as recognition accuracy deteriorated, more time and user interaction were required to complete a request but breakdown was rarely catastrophic.

Several error correction strategies were available to the user if misrecognition resulted in a semantically meaningful sentence and a corresponding erroneous action. Put That There maintained a short history of location and attributes for each ship, allowing the user to invoke the *undo* command. If a ship was created with the wrong attributes such as being green instead of the requested yellow, the *change* command allowed for repairs without the need to start the transaction over, e.g., "Change that to yellow."

Another feature of Put That There was dynamic vocabulary management. The *name* command allowed the user to name a ship and thenceforth refer to that ship by name. Naming was accomplished by putting the recognizer into a selective training mode to create a new template when the user spoke the name and associating this template with the ship in the object database. Dynamic training was facilitated by the fact that the recognizer built vocabulary templates from a single training pass; otherwise, the user might have been required to speak the name several times.

The speaker-dependent aspect of the speech recognizer was used to advantage when a multi-user version of the application was desired. The outputs of two microphones were mixed, the vocabulary size was trimmed to occupy only half of the available template memory, and each user trained his own version of each word. As long as the two operators did not speak simultaneously, recognition proceeded as normal. Since the application knew which user had created an object, this information was added to the database to prevent unauthorized modification of one user's ships by the other user.

Put That There is noteworthy as one of the earliest user interfaces employing either multimodal input or conversational techniques using voice input and output. Although the application itself was only marginally relevant to real command and control systems, the user interaction techniques are applicable to a

variety of voice interfaces. Subsequent projects, most notably Conversational Desktop (described in Chapter 12), focused on the parsing and error management techniques initiated with Put That There.

SUMMARY

This chapter discussed incorporating speech recognition into interactive applications. There are many approaches to recognition, and different classes of speech recognizers are amenable to varying styles of applications. Recognition may be employed when no other input channel is available, as an adjunct to existing input channels (keyboard and mouse), or as a direct substitute for the keyboard for text entry. Each of these application areas has different recognition requirements and derives unique benefits from the use of voice input.

Poor recognition accuracy is the dominant concern in designing interaction techniques to employ speech input. Insertion, rejection, and substitution errors all degrade performance but in different ways. Errors can be minimized by careful choice of vocabulary and microphone placement and by choosing cooperative subjects, but these factors may not apply to real-world deployment of recognition technology. As recognition accuracy improves, errors inevitably occur and can be managed through user participation in a variety of confirmation and repair strategies; however, it must be stressed that some interaction techniques will always be required to ensure accurate communication. This chapter discussed the use of language and task constraints in the automatic detection of recognition errors; we will return to the topics of language understanding and dialogue structure in Chapter 9.

This chapter's discussion of error recovery implies that developing recognition-based applications is much more demanding than simply adding a new input device to existing applications. Xspeak was offered as an example of a meta-application, controlling the window system, and thereby selecting an application to receive keyboard events rather than modifying the applications themselves. Put That There was described as an example of simple parsing techniques and spoken dialogue with the user to resolve ambiguities. Although Xspeak illustrates an attempt to closely match the specific assets of speech recognition to the range of tasks performed at the workstation, the experience of its users also demonstrate the frustrations associated with using recognition in daily work.

Higher Levels of Linguistic Knowledge

Chapter 1 presented an analysis of human voice communication as a hierarchical stack of linguistic layers. In this schema (depicted in Figure 1.1), each layer of the stack is progressively more removed from the acoustical events associated with speech. The higher layers pertain to the meaning and intention of an utterance rather than to the actual sounds used to convey individual words. As proposed in Chapter 1, analyzing conversation into layers is useful for computational purposes because each layer is likely to necessitate its own representation.

The discussion of speech recognition and synthesis technologies emphasized the lower linguistic layers of communication, i.e., speaking or recognizing words. Syntactic and semantic constraints were introduced as aids to connected speech recognition and influences on intonation and stress. The previous chapter focused on interactive techniques to recover from speech recognition errors with the goal of properly decoding the words in an input utterance. From the user's perspective, speech recognition is not about identifying words; rather, it is a means whereby a computer system understands and acts upon the user's desires. Performance and utility of speech systems are ultimately judged against the highest layers of speech understanding. To this end, this chapter presents representations of syntactic, semantic, pragmatic, and discourse structure that serve the reader as an introduction to topics in natural language processing. The chapter includes several case studies to illustrate the wide range of conversational interactions made possible by models of discourse.

SYNTAX

Syntax is structure in language imposed by the limited number of ways in which words may be combined in various linguistic roles. Words have different characteristics in a language. Nouns refer to persons, places, concepts, and sensations, while verbs describe actions performed by or on nouns. Adjectives modify nouns, qualifying or quantifying them, and adverbs similarly modify verbs by further specifying the action.

Syntactic structure groups words into phrases and phrases into sentences and describes the relationships between components at each level of structure. At a low level, syntax constrains the gender and number of adjectives and pronouns to agree with the noun to which they refer. The pronoun in “Jane ate his candy bar” is ill-formed if the candy bar belongs to Jane. In many languages the gender of nouns is more significant than in English, and adjectives exhibit different forms for each. For example, in Spanish “el rey blanco” (the white king) and “la reina blanca” (the white queen) illustrate the change in form of the adjective (“blanco”) as well as the determiner (“el”/“la”) depending on whether they modify a masculine or a feminine noun.

Conversational systems may utilize knowledge of syntax while trying to understand a user’s request using speech recognition; because syntax limits the possible combinations of words, perplexity can be minimized. Syntax must also be incorporated into language generation or else the system’s replies may be incoherent or less intelligible due to grammatical errors. For a conversational system to make computational use of syntax, two components are required: a grammar and a parser.¹ A **grammar** is a formal specification of language structure; it defines how words can be combined into syntactic units and how these units can be used to build sentences. A **parser** analyzes a sentence to determine its underlying structure as defined by the grammar. The following sections describe several grammatical representations and then present some basic parsing techniques.

Syntactic Structure and Grammars

Syntax provides structure in a language by constraining the manner in which words can be combined according to a regular set of rules. In this section we consider representations of syntactic structure and formalisms for expressing the rules that comprise a grammar. We can represent the layers of syntactic structure as a **tree**, starting by representing the complete sentence as the root node; this is similar to the technique of diagramming sentences, which many of us learned in elementary school. For example, the most simple sentences contain only a noun

¹Language structure can also be represented probabilistically, e.g., using Hidden Markov Models. Although such an approach is adequate for encoding constraints to aid speech recognition, it is less well suited for representing the grouping of words into syntactic components.

(the subject) and a verb and can be represented as in Figure 9.1, e.g., “Cats purr.” Trees capture the order and grouping of words and also indicate the unique label (noun, verb, etc.) associated with each word.

To generalize to more complex sentences we must introduce the concepts of **noun phrase** and **verb phrase**. A noun phrase is a group of one or more words that act as a noun in a sentence. A verb phrase similarly allows multiple words to act as the verb. Because they can contain multiple component words, these phrases introduce another level of structure to our tree representation. A sentence such as “Orange cats purr loudly.” is represented as in Figure 9.2. The branching of this tree structure indicates that the adjective “orange” modifies the noun “cats,” while the adverb “loudly” is likewise related to the verb “purr.”

While trees such as these examples convey the structure of particular sentences or classes of sentences, they do not tell us how to derive the structure for a

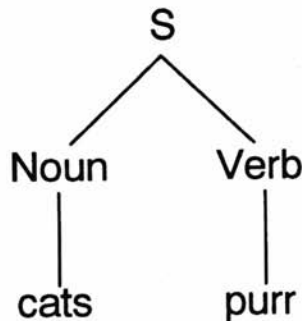


Figure 9.1. The figure shows the root node representing the entire sentence above two leaf nodes: one for the noun and one for the verb.

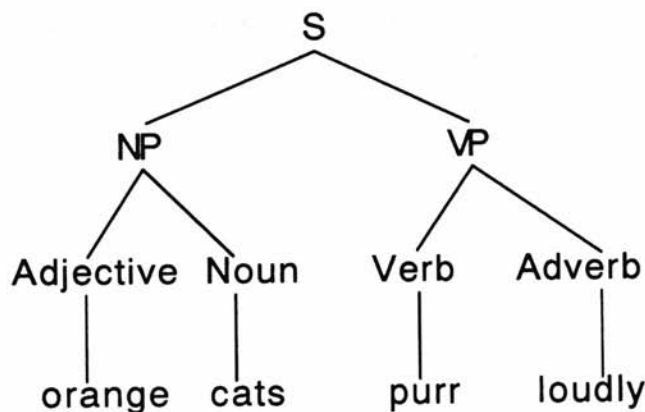


Figure 9.2. The tree representation expanded to account for a noun phrase and a verb phrase.

given sentence. Such analysis requires a set of rules that can describe all sentences of a language. A formal set of rules is a **grammar**; a complete grammar specifies the rules for all syntactic structures in a language. Grammars can be specified in several ways.

A **context-free grammar** describes syntactic structures as a series of **rewrite rules**. Each rule defines how a single symbol (on the left side of the rule) can be decomposed into one or more components (the right side of the rule). The grammar is “context free” because the symbol on the left appears in isolation, i.e., removed from the context of the other symbols. Although this constraint limits the range of sentences that can be expressed by a context-free grammar, it is nonetheless quite powerful and frequently used.

Using a context-free grammar, a sentence is broken down step by step into its constituent components through successive invocation of the rewrite rules. The symbols on the left are **nonterminal**, while those on the right may be either nonterminal or **terminal**; terminal symbols are individual words. Successive application of rules reduces all nonterminal symbols to terminals.

A grammar specifying the two sentences previously shown in tree form might resemble that shown in Figure 9.3. The first of these rules ($S \leftarrow NP VP$) specifies that a sentence “S” consists of a noun phrase and a verb phrase. The second rule defines a noun phrase as a single noun (“cats”) or an adjective and a noun (“orange cats”); the vertical bar symbol means “or.” Although this grammar describes both “Orange cats purr loudly” and “Cats purr,” it excludes many other sentence structures such as “Purr!” (no explicit NP), and “Large orange cats purr loudly” (multiple adjectives). Note that each rule breaks the sentence into the same component parts as appear on the branches of the tree representation.

We can include the first of these exceptions by modifying the first rule to read.

$$S \leftarrow NP VP \mid VP$$

The new version of the rule specifies that a sentence can consist of a noun phrase followed by a verb phrase or just a verb phrase in isolation. A minor modification of the second rule allows an arbitrary number of adjectives to precede the noun.

$$NP \leftarrow N \mid Adj NP$$

$$S \leftarrow NP VP$$

$$NP \leftarrow N \mid Adj N$$

$$VP \leftarrow V \mid Adverb V$$

Figure 9.3. A simple context-free grammar. The symbols S, NP, and VP are nonterminal, whereas the rest are terminal and correspond to individual words.

Context-free grammars can describe many sentences in a natural language such as English; they also specify most programming languages in their entirety. Note that the abbreviated discussion of context-free grammars presented here has avoided some of the issues of gender, number, and case agreement mentioned earlier. For example, the modified verb phrase rule just described allows “Purr!” which is a valid imperative sentence, as well as “Purrs,” which is incomplete.

A second powerful representation for grammars is the **transition network** or a slightly more general variation, the **recursive transition network, RTN**. A simple transition network consists of nodes and arcs as shown in Figure 9.4. Each arc is labelled with a word category, and each node is a state in the syntactic analysis of a sentence. Starting at the beginning of the sentence, each word is compared with the label on the arc from the initial state, **S**. If the word and label match, analysis transitions to the next node and word and label comparison continue as before. The last arc is labelled **pop** and always succeeds, indicating completion of the network.

This linear network does not convey the syntactic relationships between the adjective-noun and adverb-verb pairs as well as do the equivalent tree or context-free grammar representations. The network can be enhanced by allowing arcs to be labelled with the names of other networks as well as individual words; with this ability, the network has become recursive. For example, Figure 9.5 illustrates a transition network describing a noun phrase as an arbitrary number of adjectives followed by a single noun. Figure 9.6 shows how the noun phrase net-

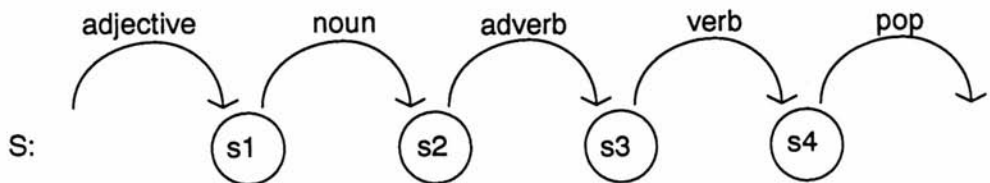


Figure 9.4. A simple transition network for sentences consisting of a single adjective, a noun, a single adverb, and a verb. The states, s1 through s4, are connected by arcs labelled with the classes of words required at each transition.

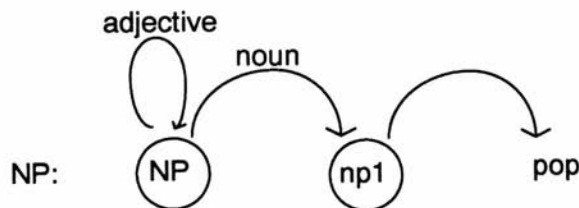


Figure 9.5. A transition network for a noun phrase allowing an arbitrary number of adjectives.

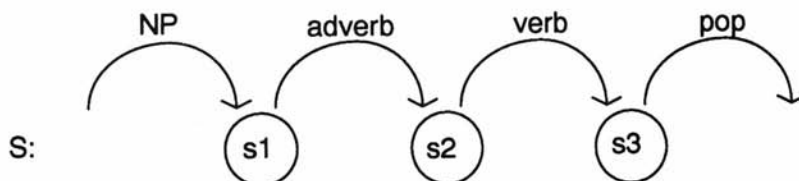


Figure 9.6. The transition network shown in Figure 9.3 modified to use the noun phrase network in Figure 9.4. This network is now recursive.

work can be incorporated as a subnetwork by modifying the simple network shown in Figure 9.4.

Let us return to the issue of agreement of gender, number, case, tense, etc. mentioned as an open issue in the discussion of context-free grammars. RTNs can manage some of the agreement issues but in a rather tedious manner. For example, we can define subnetworks for a singular noun phrase as well as for a plural noun phrase, restricting the number of the constituents of each, and then pair these with verb phrases of the appropriate number as shown in Figure 9.7. But a more succinct method is to return to Figure 9.6 and apply a condition to the arc labelled “verb” from state **s2** to state **s3**. This condition stipulates that the number (singular or plural) of the verb satisfying this arc must equal the number of NP satisfying the arc from **S** to **s1**. Although we cannot use conditional transitions within the constraints of the RTN formalism, we can extend the RTN to an **augmented transition network**, or **ATN**. Two features of ATNs are the conditional transition just mentioned and the ability to save information during transitions.

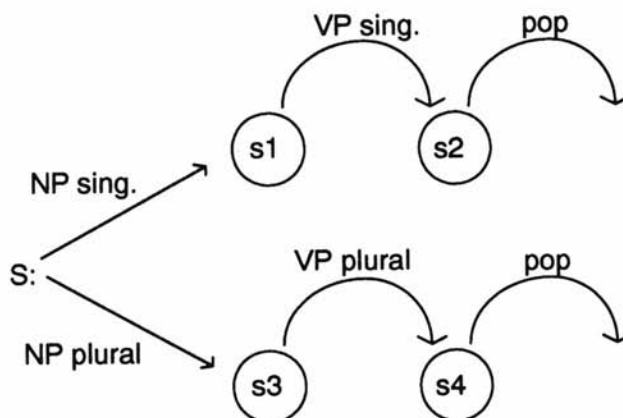


Figure 9.7. One way of representing number agreement between subject and verb using RTNs.

Parsers

We have just described the various means of representing or specifying the syntactic structure of sentences in a language. It is the task of a **parser** to determine the actual structure of a specific sentence. In its most simple form, a parser takes a sequence of words as input and given the constraints of grammar determines whether the sequence is a valid sentence. By itself this tells us little: only whether the sentence is well-formed. A more complete parser also produces a representation of valid input, which can be employed in higher layers of sentence meaning analysis.

A parser can operate in a **top-down** or **bottom-up** manner to relate the set grammatical rules to the actual structure of the given input. A top-down parser that uses rewrite rules starts with the most general rule describing sentences, i.e., that rule for which the whole sentence “S” is the symbol on the left side of the rule. Applying rules breaks the sentence into smaller components, that eventually become terminal symbols, i.e., specific words in the rewrite rules. For example, using a context-free grammar, a top-down parse of the sentence “Orange cats purr loudly” would first identify the sentence as containing a noun phrase and a verb phrase. Then the noun phrase rule, specifying a NP as some number of adjectives followed by a noun, would be selected, and as a result the parser would identify “orange” as an adjective and “cats” as the noun. At this point, the parser would analyze the remainder of the sentence by identifying the verb phrase, i.e., the right side of the tree from Figure 9.2.

The RTN version of this parse is nearly identical because the top-down parser operates from left to right within a level of syntactic structure. The first constituent of the highest RTN from Figure 9.6 is the NP; this causes the parser to invoke the NP RTN (see Figure 9.5), which allows a NP to begin with either an adjective or a noun. After traversing the adjective arc (for “orange”) the parser is still in the initial state. The next word, “cats,” is a noun leading to state **np1**, which is followed by the pop arc that returns to the RTN describing the main sentence. At this point the VP RTN can be applied.

Top-down parsing starts with the constituent units such as NP and VP and attempts to match them against incrementally smaller units and eventually words. By contrast, bottom-up parsing begins with the individual words of a sentence and tries to match each word against the grammar’s smallest constituent units. These units are, in turn, matched against higher units until the structure constitutes an entire sentence. The bottom-up parser consults a lexicon to identify the syntactic sense of each word. To continue with the example sentence, the first word “orange”, is taken to be an adjective from the lexicon. The second word is a noun. These can be matched to a NP. Similarly, “purr loudly” can be matched to a VP in the form of a verb followed by an adverb. A NP followed by a VP makes a complete sentence and the parse is finished.

In the course of parsing, **backtracking** may be required due to ambiguous sentence structure. “Orange” could be taken as a noun (a fruit) instead of an adjective in which case the bottom-up parse would hypothesize that the NP consisted of the single word “orange” (the simplest NP consists of a single noun). But then

“cats” would have to be taken as a NP too since “cat” is also a noun. At this point the parse would fail since there is no sentence structure containing two adjacent NPs in this grammar. A bottom-up parser keeps track of each decision in a data structure called a **chart** and uses this to backtrack to a previous state. For the example we have been pursuing, the parse would return to the decision to classify “orange” as a noun, classify it instead as an adjective, and proceed smoothly to complete the sentence.

Both top-down and bottom-up parsers have advantages and disadvantages. Because a top-down parser starts from a syntactic constituent and maps it to individual words, it never considers word categories in grammatically incorrect sentence positions. On the other hand, the top-down parser may process many rules before it ever considers an actual word, and it may repeatedly parse the same segment by backtracking further than necessary. A bottom-up parser avoids repetition; once words are successfully matched to a higher syntactic category they are never reconsidered. But as previously discussed, the bottom-up parser may need to consider multiple senses of each word in the process of matching it to constituent structures. These considerations have led to hybrid parsers, which are top-down in part and bottom-up in part as well.

The parsing process just described either succeeds or fails in its attempt to analyze a sentence. Success indicates that the series of words presented to the parser is a valid English sentence, and failure implies the opposite. But this conclusion alone is of little use to a language understanding system; a practical parser must also produce a computational representation of the syntactic structure and its composition from individual words. For example, if the user says “Delete all files,” a language understanding system needs to know which command the user uttered as well as its object; this specific information is more valuable than knowing the NP and VP structure of the sentence. This information is conveyed by recording the actual word that triggers each rule in a context-free grammar or each transition in an ATN. The parse then produces not only the success or failure indication, but more importantly a representation of the sentence structure and how the words of the sentence satisfy this structure.

The discussion so far has assumed that either a sentence was valid in which case the representation is desired, or the input did not form a valid sentence. But for voice interactions additional flexibility is desirable. If the user speaks a 10 word sentence and one of the words is not recognized, it is appropriate to detect this and employ some of the error correction techniques described in Chapter 8. Even if a recognition error does not occur, we often speak utterances that are only partial sentences. A **robust** parser should be able to cope with such fragmentary or ill-formed input, but most natural language processing work to date avoids this issue.

SEMANTICS

Semantics is the realm of **meaning**, in particular the meaning of a sentence considered in isolation from other sentences. A sentence has meaning in terms of

how it relates to the world and objects in it; the sentence stands for some concept or real entity. Meaning is what differentiates nonsense utterances of arbitrary words from rational sentences, and much of the study of semantics is devoted to representations of knowledge about the world.

Semantics begins with meaning at the lexical level; we cannot understand a sentence unless we know the meaning of each word in the sentence. Further, most words have multiple meanings or **senses**, and we must choose among these senses to understand the sentence as a whole. For example, “green” when used as an adjective usually denotes color (“green leaves”) but also has another sense in which it means novice or naive (“a green first baseman”). These two senses may be differentiated by considering the noun that “green” modifies. Experience is a concept applied to sentient beings not plants so the sense of “novice” is appropriate only when green modifies a noun that corresponds to a person. One word in a sentence may constrain the interpretation of other words in the sentence. Note the interplay between syntax and semantics: syntax reveals that “green” is used as an adjective and indicates which noun “green” modifies, from which we can then determine the sense in which “green” is being used.

The intended senses of words in a phrase or sentence are mutually constrained; this is called **selectional restriction**. Our knowledge of the attributes of objects in the world helps us identify the sense in which a word is being used to describe an object as we did in the example in the previous paragraph. In addition to objects, i.e., noun phrases, we also apply knowledge about how objects interact, i.e., verb phrases, while selecting word sense.

Case grammar attempts to enumerate the cases, or roles, a noun phrase can take with respect to a verb. One case is AGENT; the agent is the instigator of an action and is usually the subject of a sentence. The object acted upon, often identified as the object of the sentence, is a case called THEME. In the sentence “John smashed up the car,” John is the agent and the car is the theme. Cases such as AT, TO, and FROM indicate location. An animate object for which an action is performed is the BENEFICIARY, while an animate object in a mental state is the EXPERIENCER. The number of cases is extensive because noun phrases can be used in many different relationships to verbs. Just as real-world knowledge helps establish the meaning of noun phrases, case grammar provides a representation of meaning derived from the roles of noun phrases in an action or state of being.

From this very brief discussion it should be apparent that we obtain cues for the cases of noun phrases from their position in a sentence. For example, subjects, or AGENTS, precede objects, or THEMES, in an active voice sentence, but this order is reversed for passive voice. Prepositions are particularly strong cues for the location cases (AT, TO, and FROM). Selection of cases is controlled by verbs in other ways as well. The verb “move,” for example, has a strong requirement for a TO location case to complete its usual meaning as in “we moved to Wyoming.” For other sentences a different case is appropriate; an EXPERIENCER completes the sentence “Her singing moved the audience.” The concept of case conveys some aspects of the meaning of noun phrases in a sentence, but in the case of a noun phrase it also depends on the meaning of the verb with which it is associated. The

presence or absence of a particular case is, in turn, a cue towards selecting the sense in which the verb is being used.

Earlier in this section we discussed how the noun selects between various senses for the adjective “green.” The “green first baseman” refers to the lack of experience by a person. How can we represent the knowledge that a “first baseman” is a kind of person to provide the selectional restriction on “green”?

Semantic networks are a powerful representation for such relationships. A semantic network is a directed graph where nodes represent an individual word or a class of words, and the links represent relationships between child nodes and their parents. A very simple form of semantic network is the **type hierarchy** in which every arc represents the “is-a-kind-of” relationship. Figure 9.8 shows a partial type hierarchy indicating that a first baseman is a baseball player, baseball players are athletes, athletes are people, people are mammals, and mammals are animate objects. The type hierarchy also indicates that reptiles are animate objects and that rattlesnakes are a particular kind of snake.

The hierarchical nature of this representation indicates that all members of a lower node possess all the characteristics of their parent node, or superclass; this concept is called **inheritance**. Inheritance is useful to express generalizations concisely. With any taxonomy of this sort it is important to select the levels of classification with care; ideally, each step down the hierarchy represents a consistent increase in specificity across the breadth of the network.

A significant limitation of this form of type hierarchy is that any node can be a child of exactly one parent node, representing a subclassing of the parent along a

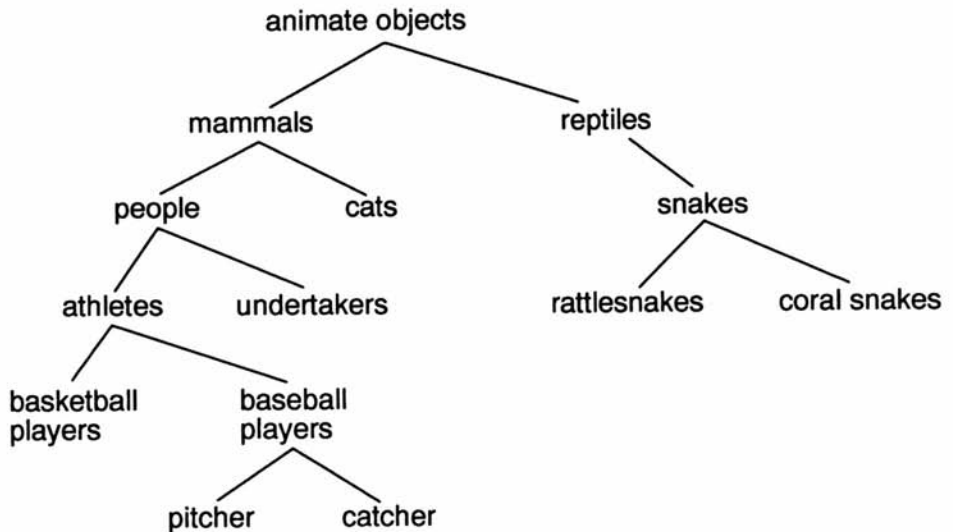


Figure 9.8. A type hierarchy. Each arc represents the is-a-kind-of relationship.

particular set of criteria. However, many conflicting subclassifications may be possible. For example, Figure 9.8 subclassed people on the basis of their profession. Figure 9.9 shows other ways of subclassing people based on sex or age. All these subclassification schemes are valid and express a set of differences among people. But these simple two-dimensional type hierarchies break down when attempting to depict multidimensional classification parameters, e.g., differentiating male basketball players from female first basemen.

Semantic networks can be used to convey concepts more general than type hierarchies and are used to represent more complex knowledge relationships. Figure 9.10 shows a sentence displayed on a semantic network in which both the type hierarchy relationships (ovals) as well as case grammar roles (squares) are drawn. Networks can be partitioned into subnetworks to keep close, specific relationships distinct from more general background knowledge or situational context, much of which may be irrelevant to the meaning of a particular sentence. Identifying the meaning of a sentence with semantic networks involves matching the sentence to a particular path or grouping within the network, so that the specific words in the sentence are mapped to their own specific meanings and the case relationship between the words becomes manifest.

PRAGMATICS

Pragmatics refers to the meaning or purpose of an utterance in a context broader than the semantic meaning just discussed. Pragmatics is about how language is used both in terms of the forms in which speech is invoked as well as in terms of its effects on the conversants. This chapter includes in the scope of pragmatics the role of the utterance in the world and among conversants (To which specific objects in the world does the utterance refer? What does the talker seek to accomplish in speaking it?); to this end it covers knowledge representation as well.

This section introduces three topics in pragmatics, beginning with knowledge representation and plan recognition, which are particularly computational views. The second topic is speech act theory, which views utterances as actions in and of themselves. Finally, a set of guidelines that regulates how much or how little we say and when we say it is introduced; this is formulated from the perspective that an utterance often contains much more meaning than is obvious from the words themselves (the semantic meaning).



Figure 9.9. Alternate type hierarchies subclassing people.

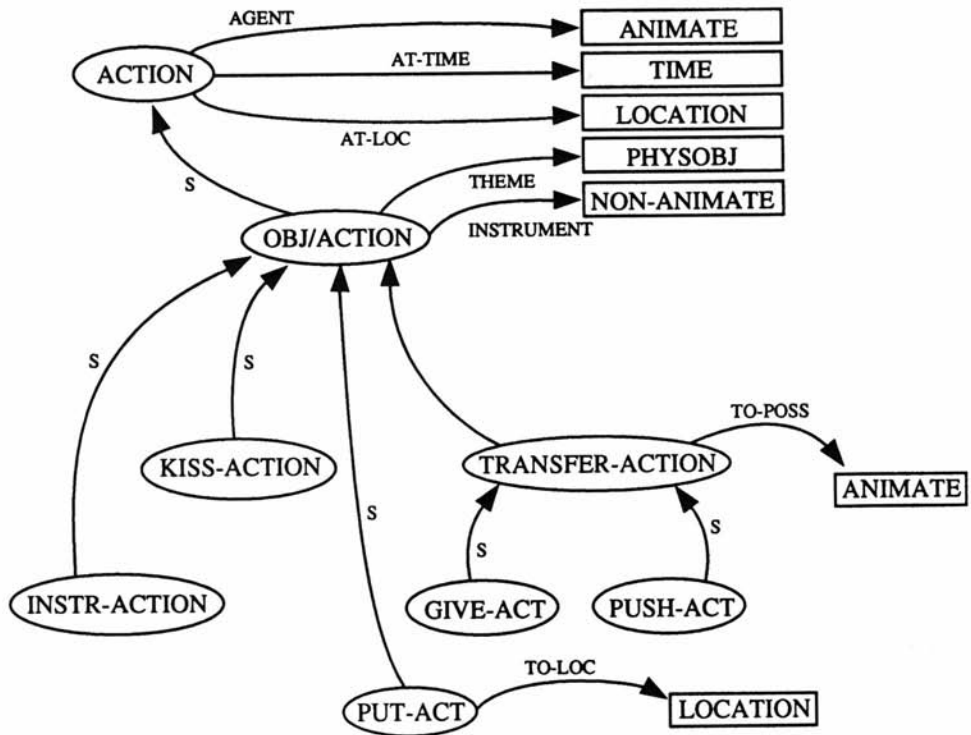


Figure 9.10. A semantic network including both a type hierarchy (ovals) and case grammar (squares) categories. After [Allen], p. 209.

Knowledge Representation

To understand what a person intends by an utterance we need to employ knowledge about the world and the objects in it being referred to by the sentence. For example, to interpret “It’s a bit too breezy in here” as a request to close the window of a car, we need to know that a breeze is motion of air, that solid objects block the flow of air, and that windows are solid objects that can be closed by some mechanism such as a crank or motor. A **knowledge representation** provides a means to store and computationally access information about the world, which may be general facts (although windows are clear, they can close an opening) or details of a specific current situation (the car window is open, and the crank will close it).

Knowledge representation systems contain two components, a knowledge base and the inference engine. A **knowledge base** is a database containing facts about the world or information about a situation, and the **inference engine** operates on the knowledge base according to a set of rules. Inference may be primarily **declarative** or **procedural**, depending on whether it stresses the knowledge database or the inference operations to draw conclu-

sions. This section touches upon several aspects of applying world knowledge to language understanding; this is meant to be just an introduction to a complex topic.

Frames are a very general representation used by many procedural language understanding systems. A frame is a loose collection of facts about a situation, some of which may be relevant to understanding a specific utterance [Minsky 1975]. Inferences are made by comparing relationships described by the frame with the semantic representation of the utterance. The important objects contained by a frame are called **roles**. For example, a frame for an airplane would likely have roles such as wings, tail, and engines; for a particular aircraft, the role of the engines might be associated with details such as the number of engines, their configuration, and manufacturer. Frames are often implemented in concert with a type hierarchy and likewise may incorporate inheritance.

Planning systems focus on the roles of actions and their grouping as a sequence to effect a change of state, or **goal**. Actions are described in terms of preconditions, the set of situations that must be true for the action to occur, and effects, or changes in the world situation as a result of execution of the action. For example, preconditions to sending a letter are writing materials and an addressee, while the effect is for the addressee to receive a written message. **Linear** plan reasoning analyzes goals in terms of sequential subgoals; the satisfaction of each subgoal yields the preconditions for the next. **Nonlinear** planning merges all subgoals into a global set of actions worrying about order of the actions only when necessary. Planning systems seek to understand utterances by recognizing the talker's plan, i.e., a sequence of actions to achieve a goal state. As a plan is likely to evolve over multiple utterances, planning systems are equally salient to discourse analysis as to semantics.

Another representation for commonly performed actions is **scripts**, which specify all the actions ordinarily required as steps toward completion of a "typical" activity [Schank and Abelson 1977]. For example, commercial air travel usually involves consulting schedules, choosing flights, making reservations, purchasing tickets, arriving at the airport, checking baggage, flying, retrieving baggage, and leaving the airport. These actions can be represented as part of a general script for traveling by public transportation (traveling by private car, bicycle, or foot must be described by a different script). Note that some steps, such as checking baggage, may be optional. Plan reasoning can be applied to scripts as well by considering each step in a script to be a precondition for the next step.

Scripts can be used to recognize the real-world activity to which an utterance refers. "I bought my ticket" could refer to any of an extensive class of activities for which admission is required (flights, plays, movies, baseball games). "I bought the ticket for my flight" can invoke the travel script, including the related concepts of departure time and destination city. Once the travel script has been recognized, other components of the script provide a framework for reasoning about the talker's activity. Once we know the ticket is for air travel, for example, we might ask "Where are you going?" or "When will you arrive?"

Speech Acts

Speech act theory considers utterances as actions in and of themselves. Much of modern speech act theory began with the work of Austin [Austin 1962]. Austin started by considering **performative** utterances, for which the act of speaking itself constitutes the action, such as:

I now pronounce you man and wife.
I hereby declare war on Germany.
I apologize for your inconvenience.

As the first two utterances indicate, these performative utterances are often of a legal or religious nature in which the talker has been granted a higher authority to speak for a society rather than merely speaking as an individual. Austin extended the notion of performative utterances with the concept that many utterances manifest a **force**, or ability to change the state of the world by their very existence, due to their impacts on the beliefs or intentions of the conversants. He identified three senses in which action is performed through an utterance.

1. The **locutionary act** is the actual utterance of a meaningful sentence.
2. The **illocutionary act** is the sense in which the utterance itself contains force to effect a change in state among conversants, e.g., the utterance may act as a promise, offer, decree, statement of commitment, or admission of guilt.
3. The **perlocutionary act** is the manner in which the utterance changes the audience by changing their beliefs or causing them to perform some action.

There have been some attempts to associate force with sentence form; some obvious examples include commanding, using the imperative form, and questioning via the interrogative form. But the mapping from force to form becomes problematic for **indirect speech acts** in which the force depends on speaking conventions. "Can you pass the salt?" is usually meant as a request not a question about the other's ability to pick up the salt shaker. In a similar vein, "It's pretty noisy in here" might be uttered as a request to turn down the radio volume.

[Searle 1976] offered a more detailed taxonomy of five classes of illocutionary action conveyed by an utterance.

1. **Representatives** are sentences by which the speaker asserts or commits to the truth of the utterance.
2. **Directives** are requests to the listener to do something. Questions are directives in that they attempt to evoke a response.
3. **Commissives**, or promises, commit the talker to future actions. Threats are also examples of commissives.
4. **Expressives** create a psychological state rather than causing a physical action. They include welcoming, apologizing, thanking, etc.
5. **Declarations** are the performative statements mentioned earlier such as declaring war or firing an employee.

The notion of speech acts is a very insightful explanation of the purpose of utterances, and much of our speech can be effectively analyzed according to Searle's taxonomy. But speech act concepts have practical limits. Speech is very social in nature, and many utterances are oriented less towards causing an action than towards establishing or maintaining a social relationship. The Coordinator, a software product that utilizes speech act theory for tracking work team assignments by requiring users to assign speech act classifications to electronic mail messages, has met with mixed acceptance in actual use [Winograd 1988, Medina-Mora *et al.* 1992]. Although utterances may be successfully classified by speech act theorists, in the actual interplay of office communication correspondents may not wish to be so direct with each other.

Conversational Implicature and Speech Acts

Conversational implicature is the principle that an utterance often contains much more meaning than the words themselves indicate directly. Imagine that two people meet on the street and the following conversation ensues.

- A: Did you do it?
B: Not again.

This conversation is largely meaningless to us; the conversants make extensive use of shared knowledge and history, and we cannot know to what action they refer. We can, however, make certain inferences about the dialogue: A and B have discussed "it" before, "it" is dominant enough in their relationship not to be ambiguous, and A assumes "it" is unlikely to be ambiguous. Another inference from the conversation is that B has done this "it" in the past and likewise understands that A knows about this prior occurrence (or most recent of several prior occurrences of "it").

To make these inferences with any degree of confidence, we resort to some implicit assumptions about the use of language. We assume that the utterances have a purpose, e.g., A and B are not total strangers but in fact have a common history. We make assumptions about shared belief, e.g., that A does expect B to know the referent for "it" and is not simply teasing B to see whether he or she can guess what is being talked about. In short, we assume an order or regularity to these utterances.

Grice's theory of conversational implicature [Grice 1975] is based on the concept that there is a set of guiding assumptions molding how a conversation is organized. Such guidelines allow us to make the inferences described above, and any deviation from the guidelines must itself have implications beyond the actual words used. Grice's **maxims of conversation** can be summarized as follows.

- **The cooperative principle:** Speak to support the accepted purpose or direction of the conversation as it is at the moment of the utterance. This is the underlying theme of all the maxims.

- **The maxim of quality:** Speak the truth; do not speak what you know to be false nor that for which you do not have adequate evidence of truth.
- **The maxim of quantity:** Be direct. Say as much as is required at the current point of the interchange, but do not say more than is required.
- **The maxim of relevance:** Make your utterances relevant and to the point.
- **The maxim of manner:** Be brief and orderly; avoid ambiguity and obscurity.

The cooperative principle defines conversation as an organized, purposeful, and efficient series of spoken exchanges, and the remaining maxims are specific techniques to support this principle. In this framework, conversants work together in a conversation, their utterances are mutually relevant, and they do not speak nonsense or seek to confuse each other.

In reality, language often is not nearly as orderly as these underlying principles seem to imply. However, in many cases when Grice's principles appear to be violated, they unify otherwise disjoint or seemingly purposeless utterances. Consider the following exchanges.

A: Did you feed the cat?

B: There's a pile of feathers at the doorstep.

B appears to violate the maxim of relevance; A wishes to know whether B has fed the cat, yet B talks about a pile of feathers. By assuming that B's reply must be relevant, we can infer that B suspects that the cat has caught a bird, and perhaps that this should substitute for the cat's dinner or that the cat should not be fed as punishment.

DISCOURSE

Discourse refers to multiple utterances over time often by different talkers. Discourse is a broad term covering several distinct areas of language understanding. The utterances in a discourse are not disjointed but rather related or connected. Discourse deals with understanding the purpose of these multiple utterances, including issues such as plans that extend across several utterances, references in one utterance to objects or concepts specified in a prior utterance, and the use of multiple utterances to comprise a single speech act. Discourse issues in conversations include the use of language to regulate the conversation's flow, the temporal distribution of contributions among its participants, and the coordination of the effective exchange of information among the participants such that all arrive at the same understanding of what was said.

Discourse issues have formed a theme of conversational interaction and feedback throughout this book. This section emphasizes how conversation is broken

into turns. The flow of turn-taking provides a collaborative environment for each talker's contributions to the conversation. Conversants maintain a common focus across turns; without this, pronouns could not refer to ideas and objects mentioned in an earlier sentence. Feedback techniques to ensure mutual understanding are discussed in the subsequent section.

Regulation of Conversation

We all know from personal experience that in a conversation the various talkers take **turns** speaking. After each turn, remarkably little time transpires before the next turn begins. Occasionally turns overlap, as one participant begins before the previous has completely finished, but it is remarkable that conversations can be as dynamic and fast paced as they are without more "stepping on each other's toes." Equally remarkable are the conversational processes for selecting a mutually agreeable topic, moving on to new topics, and returning to a previous topic.

Conversation is rich in social conventions that invite discourse and most utterances occur in a discourse context [Goffman 1981]. Turns in conversations are regulated and ordered to allow a chain of utterances to refer to a single topic; often subsequent utterances can be understood only in the context of the earlier portions of the discourse. The most simple example of this dependency is the **adjacency pair** [Sacks *et al.* 1974], in which something is presented or proposed in the first utterance and responded to, accepted, or rejected in the rejoinder. For example:

A: I brought in the mail.

B: Thank you.

A: How much does this cost?

B: Two dollars.

Note that the second utterance in the pair, which brings the pair to some form of closure, has little clarity of its own outside of the adjacency pair.

Where applicable, adjacency pairing simplifies the question of how the listener knows when the talker's turn is over as the listener must clearly wait for the proposition to be presented in the first member of the pair. Although it may be suggested that all conversations can be reduced to sets of adjacency pairs possibly with inserted sequences of other adjacency pairs between the first and second member of a pair, most conversations are more complex and resist this analysis.

In the absence of simple pairs, it is harder to specify when one turn has ended and another talker may begin a new turn. What constitutes a turn? How does the talker signal this to the listener? Turns are often composed of one or more syntactically or semantically meaningful units. These units may correspond to sentences, but they are equally likely to be smaller phrase-like units; fluent conversation often contains incomplete sentences. One appropriate unit is the **breath group**, or the string of words between catching one's breath, which usually expresses one or more coherent thoughts.

For detecting turn boundaries, the problem lies with the “one or more” of the preceding paragraph. If one talker spoke and the second always picked up when the first stopped for breath, turn taking would be more predictable. But the talker may continue to “hold the floor” for multiple utterances or the listener may interrupt before the talker has even finished or otherwise signal so that the talker modifies the utterance even as it is being produced.

The time between turns is too short (often shorter than pauses within a turn) to believe that the listener simply waits to hear if the talker has more to say. [Duncan 1974, Duncan 1972] analyzed a number of conversations and suggested the following indicators in addition to the completion of a syntactic unit of subject and predicate by which the talker can signal the end of a turn.

- **Intonation:** A level or falling pitch at the end of a sentence indicates termination of the thought being expressed.
- **Syllable lengthening:** The final syllable, or more correctly the final stressed syllable at the end of a turn, is longer than it would be otherwise. Duncan refers to this as “drawl.”
- **Gesture:** Termination of a hand gesture while speaking acts as a cue that the accompanying turn is drawing to a close.
- **Key phrases:** Certain phrases such as “you know . . .” at the end of a syntactic unit are often spoken at turn termination.
- **Visual attention:** Talkers often avert their gaze from the listener during an utterance so looking back to the listener could cue the end of a turn.

The completion of each syntactic unit or phrase is a possible end of the turn. If the talker indicates termination by cues such as those just listed, this invites the listener to take a turn. If the current talker desires to continue the turn, i.e., to present a subsequent phrase, the end-of-turn cues can be avoided or more strongly, the opposite behavior can be invoked such as using a rising intonation or beginning a hand gesture.

Conversation does not always break into turns cleanly. Sometimes, either deliberately or accidentally, the listener may **interrupt**, i.e., begin speaking before the other has finished. Interruption is usually dealt with effectively in conversation: Often one party will back off and quickly cease speaking, in which case whichever party is then the talker tends to repeat or summarize what was said during the period of overlap to insure that it was heard correctly. During overlap, the conversants have not yet resolved who should have a turn; one party may attempt to assert control by emphasis such as speaking more loudly or with increased pitch range or lengthened syllables.

Speech may be used by the listener in a manner that initially seems to be a short turn or an interruption but does not really take a turn away from the talker. **Back channels** refer to a number of behaviors whereby the listeners give feedback to the talker [Yngve 1970]. They include paraverbal utterances (“Hmmm,” “Uh-huh”), completing the other’s sentence or offering a paraphrase of it, short interjections (“Of course,” “You don’t say?”), head nods, and various facial expressions.

Back channels are a cooperative mechanism; listener feedback indicates what is known or accepted so that the talker can continue the exposition with confidence. Back channels make for more productive conversation. For example, in an experiment by [Kraut *et al.* 1982, Kraut and Lewis 1984], subjects described scenes from a film to a listener who attempted to identify the film. If the listener who was out of sight could not speak back, it took longer for the talker to adequately describe the scene. Even an eavesdropper who could never be heard benefited from the listener's back channel utterances but not as much as the listener did. This suggests that some aspects of back channel cooperation produce generally "better" utterances from the talker, while other aspects of performance improvement are specific to the participation of the back channel provider.

Discourse Focus

Back channels are just one aspect of collaborative behavior in conversation. In the course of speaking conversants change or agree upon the topic of conversation, refer back to previous topics, and reaffirm their basis of mutual belief upon which they can build successful references to world knowledge, either generic or specific and situational. The discussion of turn taking emphasized pairs or short sequences of talk. We now turn our attention to longer conversations with perhaps quite a few turns.

At any moment in coherent discourse the conversants usually agree on what is being discussed. From time to time, the topic of conversation changes. The group of sequential utterances that refers to the same topic is a **discourse segment**. Transitions between discourse segments are often indicated by **cue phrases** such as "By the way . . .," "Yes, but . . .," and "Well. . . ." Throughout a discourse segment, all utterances refer to the same topic or noun phrase; this is the **focus** or **center** of the discourse segment.

Identification of the focus of a discourse segment is required to resolve **reference**, which arises from several sources. **Deixis** is the reference of certain pronouns, such as "this" and "those" that point at something either physically or conceptually. **Anaphora** is the reference implied by pronouns such as "he" or "their." The entity referred to by deixis or anaphora is the **referent**. The referent corresponds to the focus of the discourse segment; changing the referent introduces a new discourse segment.

A discourse segment can be interrupted by the introduction of a new discourse segment, and the original discourse segment can be returned to. For example, consider the discourse fragment.

- A: California gets so green in the winter, I love it!
B: Seattle gets a lot of rain in the winter too, but not much sun.
A: It's a nice city, but you should check out Hawaii if you want wonderful winter weather.
B: Last winter we went hiking there.
A: Sometimes it gets a spell of rain in February.
B: But it's not as bad as back there! It's so dreary all winter.

In the first sentence, speaker A references "California." Speaker B then introduces a new reference "Seattle." Speaker A refers back to Seattle at the beginning of the next utterance but then introduces a third focus "Hawaii" using the cue phrase "but." The next two utterances then refer to Hawaii as well. In the last utterance, speaker B jumps back to the focus of "Seattle" without needing to further specify the pronoun. How is this accomplished without further negotiation?

[Grosz and Sidner 1986] proposed a discourse model differentiating the **attentional structure** that specifies the target of reference from the **intentional structure** that is roughly the pragmatic purpose of the discourse segment. They suggested a **stack** model for the attentional structure. A stack is a data representation in which items are put on ("pushed") and removed ("popped") from the top so that the most recently pushed item is always the one that gets popped. Figure 9.11 shows the stack progressing during the course of the example discourse. In the last snapshot, the top focus "Hawaii" has been popped, leaving "Seattle" exposed as the prime candidate for reference.

This model suggests that once popped, an object cannot be referred to again by a pronoun without being specifically introduced as a new focus so that it appears on the top of the stack again. But this is not entirely true, revealing that the model although powerful is incomplete. Speaker B might say, after a pause and somewhat longingly, "It was so nice there, what a great place to walk" referring back to Hawaii. Somehow the conversants would shift back to this focus, aided by the tense shift in B's utterances.

How do we know when a new discourse segment is introduced? In addition to the cue phrases mentioned above, the way reference is used signals a new discourse segment. Grosz, Joshi, and Weinstein use the term **backward-looking center** to refer to the entity in the current utterance that refers to the previous utterance [Grosz *et al.* 1983]. They suggested that as long as the center of the current utterance is the same as that of the preceding utterance, a pronoun should be used. If a pronoun is not used, this might suggest that a new discourse segment is being introduced. As the focus of conversation changes, the new topic may be introduced explicitly as the theme of a sentence, or it may be selected by reference from a series of things that have already been talked about (i.e., past backward-looking centers). Which of the possible backward-looking centers is selected depends on their ordering, which is dominated by recency.

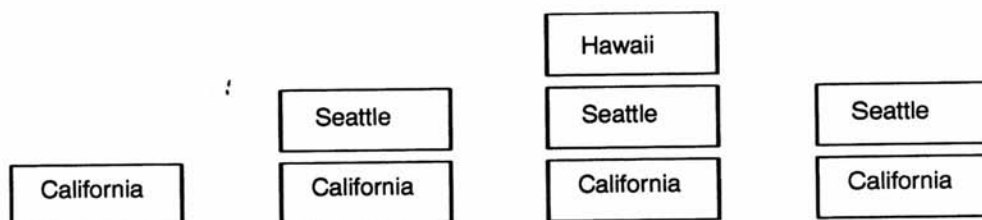


Figure 9.11. A series of snapshots of a stack model of the shift in focus of the weather discourse. Two new topics are introduced. The last utterance pops the top topic Hawaii to refer back to Seattle.

Intonation is a strong cue to shifts in focus. New information, i.e., potential new centers, are usually stressed more than old information. Intonational cues can also be used to override the default ordering of centers.

- 1: John hit Bill and then he called the police.
- 2: John hit Bill and then HE called the police.

In sentence one John calls the police; in sentence two Bill does. Ordinarily we would expect that “he” refers back to John, but if this word is emphasized as in sentence two it indicates to choose an alternate center, i.e., Bill.

Focus management requires the agreement of all participants in a conversation—if one party introduces a new topic without the appropriate cues, then other conversants do not receive the intended message. Conversation is a cooperative process. Another aspect of collaborative conversational behavior relates to synchronizing mutual beliefs. Clark *et al.* identify a number of conversational moves to make sure that both conversants agree to the identity of the current focus [Clark and Marshall 1981, Clark and Schaefer 1989, Clark and Wilkes-Gibbs 1986]. Clark and Brennan [Clark and Brennan 1990] introduce the concept of **grounding** to describe the process whereby we ensure that the listener understands our utterances as we intend them to be understood and that we agree that they stand for the same entities in the world.

Although much of conversation is purposeful and can be categorized as an attempt at communicating a concept or performing an action, conversation also serves a social nature. Part of the feedback provided by back channels, for example, informs the talker “I am paying attention to you. I am listening. I value your contribution.” Sometimes talk exists as much to fill a communication space between people who feel awkward in silence as it does to change the other conversant’s opinion or affect changes in the physical world [Goffman 1981].

CASE STUDIES

This section presents case studies of several projects that attempted to maintain interactive conversations utilizing the aspects of higher-level linguistic knowledge described in this chapter. Although only fragmentary use was made of the formalisms just described, these case studies offer some evidence of the potential of the topics in this chapter for enabling more sophisticated conversational interaction. Syntactic and semantic knowledge can be used to detect speech recognition errors and to determine the meaning of an utterance. Pragmatics relates an utterance to the larger world situation, and discourse structure helps cue appropriate responses.

Grunt

Grunt was an experiment that explored the utility of a discourse model in isolation from other linguistic knowledge [Schmandt 1988]. Grunt attempted to main-

tain a conversation by listening to the user, but without the use of word recognition. Without word recognition, the ability to negotiate conversational roles or topics was limited, so Grunt was engineered around the predefined, specific task of giving driving directions between two known points. The goal of the system was to employ acoustic feedback (or lack of it) from the user to improve his or her ability to understand and transcribe the directions. Grunt was complementary to its contemporary, Direction Assistance (described in Chapter 6), which allowed its user to choose a source and destination but did not offer any conversational flow control while reciting driving directions.

Grunt's discourse model was developed after recording and analyzing conversations between two people who could not observe one another, in which one was assigned the role of writing down the directions spoken by the other. Grunt's task was to present segments of the route as a series of turns, and the user was encouraged to engage in natural discourse behavior for conversational flow control. Subjects spoke to Grunt over the telephone without having been given any hints about its capabilities before the interaction; they had been told that they would hear driving directions to a bakery on the other side of town and that they were requested to write down the route.

Initially Grunt employed a very simple discourse model which was then elaborated over the duration of the project. Grunt would speak a sentence and then listen for a reply, relying on back channel utterances from the listener to speed up the conversation. Since there is great variability in how fast people transcribe directions, if Grunt waited after speaking sufficient time for the slowest writers to finish most others would become frustrated. Hence the need for flow control. A user's spoken response was detected by the change in audio level; at the end of the reply, Grunt would proceed to the next step of the bakery directions. If the user said nothing, Grunt would wait a rather long time and then proceed with the directions. To encourage back channel behavior by the user, Grunt would occasionally query a mute listener "Are you there?" (subjects always responded to the question) and offer hints such as "I'm listening to you too."² Such exchanges, called **channel checking** in [Hayes and Reddy 1983], are not uncommon during telephone conversations in which one conversant speaks much more than the other.

This model might have worked if the listener consistently and completely understood the directions when they were first spoken but this was not the case. People sometimes misunderstand each other, and successful communication was further hampered because the listener had to cope with Grunt's text-to-speech synthesis. When we cannot comprehend each other's directions, we ask for clarification; how could Grunt differentiate these requests from back-channel utterances? Based on the observation that back channel acknowledgment responses tended to be very short ("uh-huh," "OK," "hmm . . ."), Grunt measured the dura-

²As further encouragement of listener participation, subjects were asked "What is your name?" and "Are you ready?" by Grunt before any directions were given to help them realize that their speech could be heard by the computer.

tion of each utterance, and if the response was longer than 800 milliseconds, it assumed that the listener was confused and had asked either a question or for repetition. In response to this “clarification” reply, Grunt said “I’ll repeat” and repeated the directions verbatim but at a slightly slower rate. Because Grunt’s assumption that the listener was confused was based on skimpy evidence, it was important to explain “I’ll repeat” so as to not risk misleading the user. For example, in the following dialog the user might insert an extra mile-long leg followed by a left turn into the directions.

Grunt: Go about one mile and take a left at the first stop sign.
 Listener: I think I know where that is.
 Grunt: Go about one mile and take a left at the first stop sign.

Although extremely simple, the discourse model described so far (see Figure 9.12) was surprisingly effective and robust. In the constrained context in which subjects were exposed to Grunt, it behaved flawlessly for about one out of five users. Observations of failure modes in which the conversation digressed from Grunt’s discourse model and its responses were therefore inappropriate resulted in several improvements.

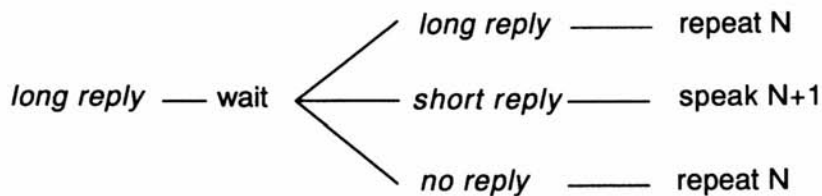


Figure 9.12. The states in Grunt’s discourse model.

The first improvement extended the discourse model to cope with synchronization responses by the listener. If the listener was having difficulty keeping pace while writing, he or she might utter “Just a minute, please” or “Hold on a second.” In normal human interactions, utterances such as these suspend the conversation, giving the requestor the floor until explicitly yielded.³ This final return of the floor is often accomplished by exactly the same short back channel style utterances that Grunt already used as a cue, e.g., “OK”. This timing cue was utilized as shown in Figure 9.13. When a long utterance was heard, Grunt no longer immediately repeated the current direction segment. Instead, it waited in anticipation of further speech to differentiate synchronization from clarification requests. If the next utterance was short, Grunt assumed that a synchronization

³Resumption of conversation after the granting of the synchronization request need not be explicitly triggered if it becomes obvious to the conversants that talk may continue. For example, if the note-taker were visible to the person speaking directions, the direction-giver could watch for the note-taker to finish writing or to look up.

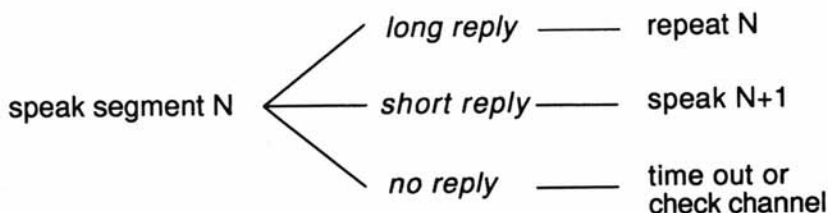


Figure 9.13. Modified Grunt behavior to detect synchronization requests.

cycle had just been completed and moved on. If the subsequent utterance was long, Grunt repeated the current segment assuming that the listener had previously asked for clarification, grown tired of waiting, and was now reiterating the request. This model thus provided for a correct response to a synchronization request and a delayed, but still correct, response to a clarification request.

The second improvement to Grunt's discourse model involved recognizing intonation to detect questions among the short utterances. Initially, Grunt erroneously treated short questions like "What?", "Where?", and "Left?" as acknowledgments. When this occurred, listeners became both annoyed and more confused since Grunt would apparently ignore their question and proceed with the next segment. With the improved model short utterances were analyzed for pitch, and those with clearly rising intonation were treated as clarification requests instead of acknowledgments (see Figure 9.14).

Toward the end of the project Grunt's discourse model was extended to include interruptions by the listener. When interrupted, Grunt would cease speaking, thus yielding the floor. The problem then was how to resume the dialog. Grunt could not know why the listener interrupted so it attempted to respond properly to any back channel acknowledgments. If an interruption was short and not a question, Grunt would repeat a few words and continue the utterance. If an interruption was longer, Grunt would wait for further input after it ceased speaking. After the listener spoke again, the current direction sequence would be repeated from the beginning as Grunt could not ascertain how much the listener had heard the first time.

Figure 9.15 depicts Grunt's final discourse model minus the interruption branches. Each of the advancements over Grunt's original, simple discourse model improved the probability that Grunt would do what the human listener expected;



Figure 9.14. Grunt analyzed short utterances to detect whether they might be questions and reacted accordingly.

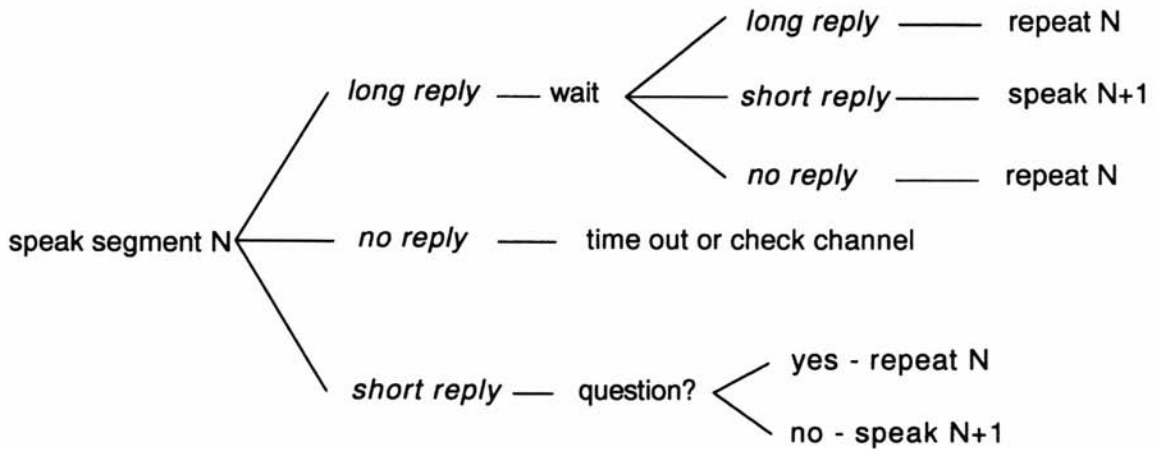


Figure 9.15. Discourse states for the final modified version of Grunt. Interruption is not shown.

this was based on observing breakdowns during its use. However, at its best, Grunt had considerable limitations and exhibited several distinct failure modes.

Summarizing or completing the talker's sentence is a back channel behavior that ordinarily moves a conversation along, and while transcribing directions many listeners invoked such very explicit flow control, but Grunt misinterpreted these as clarification requests. A related behavior is to recite word by word what one is writing as timing feedback to the direction-giver. Grunt interpreted these as a series of short acknowledgement utterances and became confused. Grunt tried to minimize the confusion resulting from false conversational moves by stating very explicitly that it was repeating or paraphrasing, which unfortunately sometimes led to wordy exchanges.

Other failure modes were caused by listeners changing their conversational behavior due to misconceptions of the discourse capabilities of computers. Almost one quarter of the participants said nothing while listening to the directions except when asked directly "Are you there?" Even when the period of time that Grunt waited during the silence between utterances was increased to the extent that interaction became exceedingly slow in the absence of acknowledgments, a significant number of listeners were simply mute, either not realizing that the computer could listen or feeling uncomfortable speaking to it. Some knowledgeable subjects believed that the computer was listening via speech recognition; based on the essentially correct understanding of recognition technology, they spoke in single word utterances, e.g., "Repeat," in order to be understood. But Grunt was not using word recognition, and these misguided attempts to cooperate broke the discourse model.

As a research project, it was intriguing to see how well Grunt could manage a discourse model in the absence of speech recognition. In a way, it was refreshing simply because so much work with recognition gets bogged down in syntactic and

semantic analysis and requires corrective discourse which, in turn, detracts from the naturalness of smooth conversation. The extent to which Grunt succeeded greatly surprised its designers, even though this success was heavily dependent upon having chosen a very limited and focused task with well-defined roles for the conversants. Nonetheless Grunt did demonstrate the power of even a minimal discourse model.

Conversational Desktop

Conversational Desktop, another Media Lab project (1984), provided an integrated voice interface to an integrated office telecommunications and message-handling environment [Schmandt *et al.* 1985, Schmandt and Arons 1986]. Conversational Desktop could place telephone calls, take voice messages, schedule meetings, and answer queries about airline itineraries; its functionality and telecommunications aspects are detailed in Chapter 12. The case study here discusses its language understanding and dialog-generation components, which utilized connected speech recognition for input and text-to-speech synthesis for output.

Conversational Desktop's conversation model provided a context in which to interpret spoken commands and also engage the user in a dialogue to correct recognition errors. Conversational Desktop's language understanding approach illustrates how syntactic, semantic, pragmatic, and discourse knowledge all contribute to conversation management. Although the contributions of this project at each layer were small and informal, their combination shows the richness of a conversational user interface.

The discourse system of the Conversational Desktop project can be thought of as a simple frame-based system. An empty frame was a data structure with fields for the different semantic classes of words in a small vocabulary. Some of these fields were filled in with specific words detected while parsing the user's input. Various procedures were then invoked to determine if sufficient information had been gathered to perform the requested action; these procedures could look to other sources of knowledge (databases) to support inferences.

In Conversational Desktop, the syntactic and semantic portions of language understanding were combined into a context-free grammar-based parser. Semantic knowledge was incorporated into the parsing process by basing the grammar on word categories more specific than the usual syntactic classes of noun phrase, verb phrase, preposition, etc. For example, Figure 9.16 illustrates a fragment of the grammar describing sentences such as "Schedule a meeting with Walter tomorrow at 10:00" or "Phone Chris and Barry." During execution, the parser recorded two categories of information when each grammar rule was applied. First, when any terminal symbols were encountered, both the word and its semantic role were stored in a frame-like structure describing the utterance. The frame had positions (roles) for the various "cases"⁴ that might be found in a sen-

⁴The cases used were ad hoc and specific to the task as opposed to the more general cases described earlier in this chapter.

```
sentence := CMD | CMD_N NAME | CMD_NT N_AND_T

CMD_N := phone | where is

CMD_NT := schedule a meeting

N_AND_T := PERSON_NAME TIME | TIME PERSON_NAME

NAME := PERSON_NAME | PERSON_NAME and PERSON_NAME

PERSON_NAME := Chris | Barry | Walter

TIME := DAY HOUR | HOUR DAY

Day := today | tomorrow | Monday | Tuesday ...
```

Figure 9.16. Conversational Desktop's parser used a context-free grammar based on semantically meaningful word categories.

tence; as the parser recognized a word fitting into a case, that word was saved in the frame as an instance for its case.

The second result of the parser's analysis was a list of additional information that would be required to complete the sentence. For example, when the parser noticed the SCHEDULE-MEETING action, it would note that a PERSON, a DAY, and a TIME were required and add these word classes to the list. Similarly, when encountering AT used in the temporal sense, Conversational Desktop would note that a DAY and a TIME were required to complete the sentence. As each of these missing classes was located later in the parsing process it was removed from the list. If this list was not empty after further pragmatic and discourse analysis, Conversational Desktop had an incomplete understanding of the user's request and engaged in dialogue to identify each item in the list.

A unique attribute of Conversational Desktop's parser was the underlying assumption that the input from connected speech recognition would be error-prone. With the use of connected recognition, not only can any individual word be misrecognized, but the possibility of insertion and rejection errors also implies that even the number of words sent to the parser can be incorrect. The goal of the parsing process was to extract whatever useful information was contained in the recognized words. In other words, it was to identify the most likely utterance spoken given the words reported by the recognizer, the syntactic and semantic constraints of the context-free grammar, and knowledge of error probabilities.

input words:	ABCD	parser candidates:	ABCD
			ABC ACD BCD . . .
			AB AC AD BC . . .
			A B C D

Figure 9.17 The Conversational Desktop attempted to parse all substrings of the input utterance. Each substring was evaluated, and those which could be parsed were candidates for selection.

Because any word reported by a speech recognizer might be a spurious insertion error, all substrings of the input words were parsed as shown in Figure 9.17. If *any* words were recognized correctly, one of the substrings would contain exactly all these words without the insertions. All of the substrings were submitted to the parser, which had to reject those strings containing grammatically correct insertions, but accept the substring that represented the original sentence minus the insertion.

Rejection errors caused incomplete sentences and resulted in fragments of syntactically correct tokens. To cope with such errors, the grammar was modified to describe sentence fragments as well as complete sentences, e.g., the word sequence “Barry Friday at two o’clock” is a well-formed fragment from a sentence about scheduling a meeting. This parsing strategy also handled those substitution errors resulting in semantically incongruous sentences; if one of the substrings contained exactly the correct words, then the resulting sentence fragment would be accepted by the parser’s rules for fragments.

These parsing strategies had to be combined because any type of error could occur in each sentence. Because of the fragment rules, multiple substrings from each input sequence would usually be accepted by the parser; the next stage of analysis selected the best of these. Evaluation of “best” was based on empirically derived weightings of the following.

- **Completeness:** Assuming that the user spoke well-formed sentences, a successfully parsed complete sentence was more likely spoken than was a sentence fragment.
- **Number of words:** A greater number of words in an input string was preferred because recognition results were more likely to be correct than erroneous. This judgment was also based on the assumption that the user spoke sentences based only on words in the recognizer’s vocabulary. If both “ABC” and “AB” could be parsed, “ABC” would receive preference from this metric.
- **Sequences of words:** Because of the difficulty of word endpoint detection, connected recognizers tend to produce runs of correct or erroneous results. If the endpoint of one word is incorrectly identified, not only is that word likely to be misrecognized but also the following word. The scoring metric favored substrings that included

consecutive tokens matching the input. Substring "AB" was favored over "AC" from input "ABC."

Once all candidate substrings had been evaluated and the best guess selected, further processing could be invoked if the user's request was not adequately specified by the recognized utterance. Pragmatic analysis consisted of resolving situational references in the context of the user's current or planned activity. For example, if the user said "Schedule both of us a meeting . . ." while on the telephone, Conversational Desktop assumed that the person at the other end of the connection should be included. Similarly, "When's my flight?" was evaluated with reference to the situational context; if asked while a call was in progress, flights to the caller's location were checked before the default, i.e., next scheduled flight. Pragmatic processing associated descriptions of events ("my flight") with known entities in databases (Boston to San Francisco flight on August 12th at 5:37 PM) so that action could be taken on behalf of the user.

At the final stage of language understanding, discourse analysis was invoked to resolve remaining ambiguities. This was a weak model of focus based on tracking the current task frame; there was no stack or other history mechanism in the model, so previous topics could not be recalled. The model of the current task allowed limited resolution of anaphora, thus supporting transactions spanning multiple sentences. For example, the user might inquire about someone's availability for a meeting: Conversational Desktop would consult schedules and suggest a time, which the user could accept by saying "Confirm it." Or having requested information about a scheduled activity, the user might command "Delete it."

Conversational Desktop also initiated dialogue. Despite the various stages of language analysis just described, errors in speech recognition often prevented complete identification of the user's request. When this happened, the list of missing roles for the sentence would not be empty, but instead it would specify missing information required to complete the request. Conversational Desktop phrased a question around the needed information but also echoed much of the request as understood so far: "At what time tomorrow do you wish to meet with Barry?" Echoing provided feedback and reassurance that the transaction was proceeding, and it could alert the user quickly when the recognizer had been severely mistaken. More importantly, revealing to the user what had been recognized allowed human detection of semantically correct recognition errors that Conversational Desktop could never detect. If, for example, the user had requested a meeting with "Harry," but this was recognized as "Barry," user intervention ("Change that to 'Harry'.") was the only recourse. Without echoing, this semantically correct error would go undetected.

Much like the research project Put That There (see Chapter 8), Conversational Desktop awaited further input after speaking a query. Except for a few special commands ("Change that" and "Cancel"), this additional input was merged with what had previously been recognized and the parsing cycle was repeated. This process continued until an action could be taken. As recognition accuracy decreased, it took longer for the user to complete a transaction, but the interaction rarely broke down completely.

SUMMARY

This chapter considered computational aspects of the higher-layer aspects of speech communication whereby words are combined into sentences and sentences into conversation and how these structures imply intentions and actions in the real world. Syntax and semantics embody the constraints whereby words are combined in certain sequences to form rational utterances. Pragmatics allows these utterances to carry force, while discourse provides a framework that structures a sequence of utterances into a coherent whole. To employ such concepts in interactive computer systems, underlying representations of each layer must be specified, and then procedures must be developed to operate on these representations.

The syntactic structure of a sentence may be represented as a tree, but a grammar is better defined by a set of context-free rules or an augmented transition network. Parsers analyze sentences in a top-down or bottom-up order as they apply these grammar rules. Semantic analysis requires representation of word meaning, and in particular it must differentiate the various senses in which a word may be used in a sentence. Case grammars represent the roles or functions a word plays in the action of a sentence, while semantic networks and type hierarchies describe the meaning of a word in terms of the other entities of which it is an instance.

Pragmatics relates the sentence both to the talker and to the situation in which the sentence is spoken; it links words to objects in the real world. This aspect of meaning, either general or specific to the current situation, requires a knowledge representation of which scripts and frames are examples. Pragmatics also considers the implicature of speech, i.e., the meaning beyond the words in a sentence such as "Do you have the time?" Speech act theory considers the utterances themselves to be actions in the world.

A number of aspects of conversation come together at the discourse layer. Discourse describes the methods and possible rules for determining who holds the floor, how talk is divided into turns, and the duration of a turn. Discourse also encompasses maintaining a focus across multiple utterances so that all conversants speak about the same topic and mutually agree on the referents for anaphora. Lastly, discourse provides a context in which conversants collaborate to discover or construct a common ground of experience and understanding so that the pragmatic implications of an utterance can be shared by all.

A case study description of Grunt illustrated that discourse models may be surprisingly useful in very constrained contexts even without word recognition. By building on accepted expectations of the flow of discourse, this simple mechanism added surprising vigor to the interface. A description of Conversational Desktop highlighted the interplay between language understanding components operating at all the communication layers described in this chapter. In this project, information gathered during parsing and further analysis was used to direct dialogue with the intent of resolving ambiguities which arose during conversational interaction.

FURTHER READING

Allen [1987] is an excellent textbook covering all aspects of the material discussed in this chapter. Winograd [1983] focuses on syntax, covering this topic in great detail. Levinson provides an excellent textbook on material included as both pragmatics and discourse in this chapter. Grosz *et al.* 1986 is a collection of readings on computational linguistics and includes many of the classic papers on plans and discourse models. Cohen *et al.* is another excellent collection of readings relevant to this chapter.

Basics of Telephones

This and the following chapters explain the technology and computer applications of telephones, much as earlier pairs of chapters explored speech coding, synthesis, and recognition. The juxtaposition of telephony with voice processing in a single volume is unusual; what is their relationship? First, the telephone is an ideal means to access interactive voice response services such as those described in Chapter 6. Second, computer-based voice mail will give a strong boost to other uses of stored voice such as those already described in Chapter 4 and to be explored again in Chapter 12. Finally, focusing on *communication*, i.e., the task rather than the technology, leads to a better appreciation of the broad intersection of speech, computers, and our everyday work lives.

This chapter describes the basic telephone operations that transport voice across a network to a remote location. Telephony is changing rapidly in ways that radically modify how we think about and use telephones now and in the future. Conventional telephones are already ubiquitous for the business traveler in industrialized countries, but the rise in personal, portable, wireless telephones is spawning entirely new ways of thinking about universal voice connectivity. The pervasiveness of telephone and computer technologies combined with the critical need to communicate in our professional lives suggest that it would be foolish to ignore the role of the telephone as a speech processing peripheral just like a speaker or microphone.

Although the telephone network was designed for voice communication, it is being increasingly used to carry data either by modem to computer or by facsimile. Technology at either end of the phone call has revealed new roles for voice connections over the telephone network: answering machines and voice mail

allow asynchronous “conversations”; echo cancellation and speakerphones enable multiparty conferences; increasingly flexible call routing services allow more freedom in associating a telephone set with a number; and the telephone accesses a growing number of information retrieval services such as those described in Chapter 6.

It is useful to abstract the major features of a telephone system into the functions of signaling and transport as these concepts transcend the implementation details of any particular network. After many years of relative stability, telephone systems are now changing rapidly with increased use of digital components and radio links to mobile telephones. While network implementation details affect the capabilities of many telephone services provided by computer systems, many of the underlying functional aspects of the network remain unchanged. Analog and digital telephone networks require unique interface hardware and consequently each may present different characteristics of the voice channel. For these reasons, this chapter describes implementations of analog and digital telephone networks in the broader context of the major functional units of any telephone system.

FUNCTIONAL OVERVIEW

This chapter considers telephones from the perspective of their role in providing transport of voice in a system that employs computers for storage, databases, applications, and user interfaces. This view is based on the minimal requirements that make effective use of voice, or any medium as an information type on a computer.

- *Transport:* The ability to move the data from one location to another. Telephones, computer networks, and package delivery services all provide a form of transport.
- *Storage:* The means to save the data for later access. Computer disks hold files, videotapes store images, and answering machines store voice messages.
- *User interface:* A means of allowing a person to retrieve and manipulate the stored information through tools such as editors. The user interface of an answering machine consists of buttons, knobs, and lights.

The telephone network provides the most commonly utilized means of transporting voice, although computer networks have also been used in some experimental systems. Since the telephone network is expansive and eventually will be uniformly digital (i.e., able to transmit speech reliably and with higher quality), researchers choose to use it in many computer scenarios.

We can separate the telephone functions for moving voice between locations into two phases. An **addressing** or **signaling** activity both specifies and establishes a connection to the desired recipient with the help of routing and signaling via the telephone network. Call setup refers to the procedures invoked from the

time the caller begins the call (usually by picking up a handset) until the time the called party answers (again usually by picking up the handset). If a connection is established (i.e., the other party answers), the **transmission** phase begins during which voice is carried over the network. Either party can terminate the call (usually by hanging up), which initiates a final signaling phase of call teardown. Note that if the called party does not answer, a call can proceed directly from the setup to teardown stages. Signaling and transmission phases exist in all telephone systems, but they are implemented differently in analog and digital telephone networks.

ANALOG TELEPHONES

Many of the telephones we use every day including almost all home and public phones are **analog**. As discussed in Chapter 3, analog devices use a continuously varying signal to convey information. Analog phones may be for a single line¹ or for multiple lines with buttons to select between lines and put a call on hold (a **key set**). These phones may have rotary dial or push buttons, which usually generate tones. The analog phone has a ringer (bell) that indicates incoming calls. The telephone is connected to the central office (CO) by wires that run along aboveground poles or in underground cables (see Figure 10.1).

The central office contains the telephone switch, which is connected to the wires from all the telephones in the neighborhood as well as to lines that connect with the remainder of the telephone network. Older telephone switches consist of various types of relays and electromechanical steppers that count the digits as they are dialed, switch lines, and ultimately complete a call. Such switches are rapidly being replaced by modern switches that are digital computers performing essentially the same job as their mechanical predecessors. For each call, the central office connects the telephone to the rest of the network by helping to complete a **circuit** that carries the voice signal between two telephones. A **circuit switch** constructs a path between two points; the circuit is dedicated to this particular path and information on the circuit automatically flows directly to the destination. By contrast, most digital computer networks are **packet switched**, allowing multiple packets of data, each with an embedded address, to share the same physical circuit or wire. The switch characteristics are invisible to the users of the telephone system, however. It is likely that someday all voice telephony will be packet switched although now it is circuit switched.

Analog telephones are connected by a pair of wires to the central office, which is where the first level of telephone switching occurs. Although telephone cable often has four wires, colored red, green, yellow, and black, only the first two are in use in most telephones. The other pair is reserved for special functions such as lighting for illuminated dials, coin collection in a pay phone, or installation of a

¹The classic American telephone from the Western Electric design is called a "2500 set."

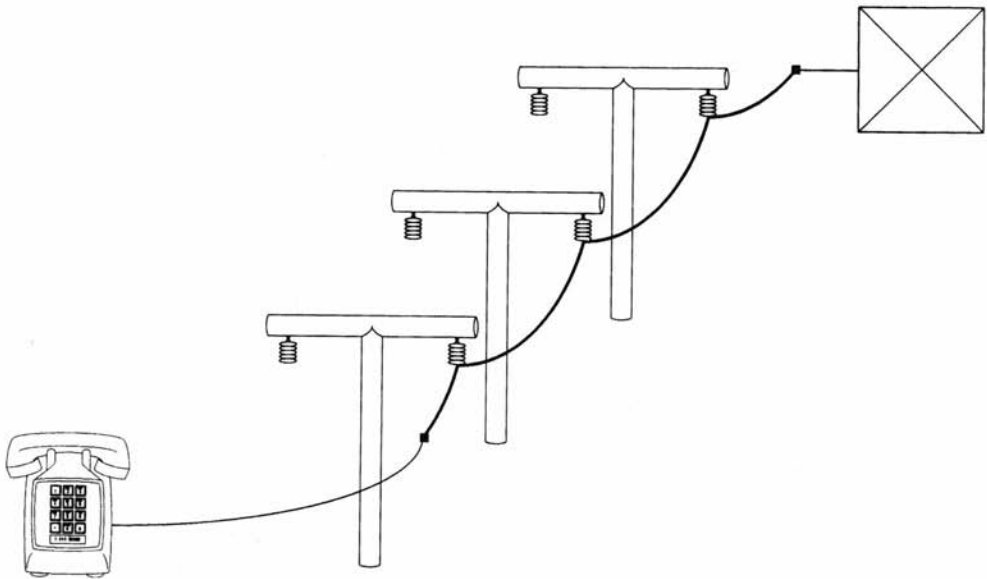


Figure 10.1. A telephone is connected to the central office by a pair of wires.

second line.² The two primary wires, called the **local loop**, are connected to a battery at the CO, which supplies a 48 volt DC voltage. The green wire is connected to the positive side of the battery and is called **tip**, while the red wire, called **ring**, is negative.³ Using a battery to power the loop allows the local telephones to continue operation in the event of a power failure and provides a constant voltage source.

Signaling

Signaling encompasses sending the dialed digits as well as alerting the destination party to an incoming call. Signaling is the transmission of control information for call management, which includes both **call setup** to establish a voice connection with another party and **call teardown** to free the resources associated with the call at its termination. With analog telephones, signaling is accomplished through a combination of varying the current in the loop and sending audible waveforms over it.

²The specifications discussed in this chapter such as color coding, voltages, frequencies, and durations of audible signals are specific to the United States and may vary in other countries. The discussion of digital telephony is based on international standards.

³These terms date back to the portions of the jacks used to complete connections via patch cords on old manual switchboards.

The wire gauge (thickness) of the wire pair, or loop, at the central office is standardized, and the entire circuit is typically balanced to provide a specified current (20 milliamps) at the telephone. These characteristics provide a uniform operating environment allowing telephones to be attached easily to different circuits. When **on hook**, the telephone presents a high impedance (10 MegaOhms) to the loop and little current flows.

When the handset is picked up, it goes **off hook** and the impedance drops to about 600 Ohms causing a larger amount of current to flow in the loop, which, in turn, signals the switch in the central office that the line seeks attention, and the switch puts a dial tone on the line. The dial tone consists of voltage variations at audible frequencies superimposed on the 48 volt battery; it is transmitted in the same manner as voice. The dial tone notifies the caller that the switch is ready to receive the digits specifying the destination address. The spectral characteristics (i.e., what frequencies are combined in what ratios) of the dial tone allow us to distinguish it from other phone sounds; this is a feedback mechanism designed for human ears.

If the dial tone is heard, the caller may then dial. The verb *dial* refers to the operation of the older style rotary switch telephone (which, incidentally, remains more prevalent in other parts of the world than in the U.S. or Canada). Each digit is entered by rotating the dial a set distance delimited by holes in the dial and a finger stop bar. As the dial is released, it is driven by a spring at a constant angular velocity, and a small switch and cam inside the dial mechanism alternately connect and disconnect the loop wires. The total of such *make/break* cycles specifies a digit. The CO switch detects these pulses as alternate presence and absence of current flowing through the loop. Both the rate of the pulses and the ratio of time during which the loop is open or closed are standardized. The necessity of waiting for the dial to return to its original position before positioning the finger for the next digit enforces a pause between digits.

The switch in the central office counts the digits and makes certain routing decisions locally. In old mechanical switches, the digit pulses actually physically moved relays to close electrical contacts; modern equipment samples the signal on the line and stores the number digitally while deciding how to route it. If the call is local and the first three digits are the same as those of the originating telephone, the call is in the same exchange and the routing of the resulting connection is entirely through the local switch.⁴ If the destination is in the same local area, the local switch may find a free trunk line to the CO switch that serves the dialed exchange. A trunk is a wire pair that may be used to carry calls associated with different numbers for different calls over time as opposed to the wires of a particular telephone, which are dedicated to that telephone exclusively. The local switch will then send the remaining four digits to the remote switch, which is

⁴Again note that this assumes the North American numbering plan, which allows three digits for an area code and seven digits for a number, of which the first three digits specify the exchange. In other parts of the world, different numbers of digits may be allocated to each of these functions.

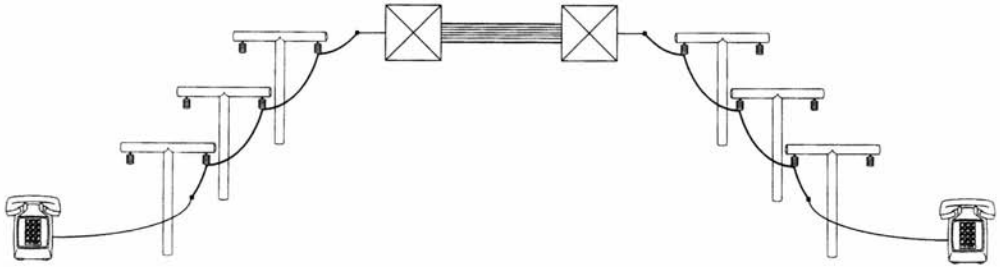


Figure 10.2. Telephones connected through trunk lines between two central offices.

responsible for the next stage of switching, i.e., connecting the trunk to the local loop of the destination telephone, as shown in Figure 10.2.

Switching becomes more complicated in long-distance dialing. In the United States long-distance numbers are usually preceded by a “1” to signal that a 10 digit number follows instead of a 7 digit local number.⁵ *Long-distance* is somewhat of a misnomer these days as the regional operating companies are no longer part of the long-distance company. If two area codes cover adjacent areas, calls may be handled by the same regional company and the service charge may or may not appear as a separate itemized billing item at the end of the month. If the destination is more distant, the call travels through the network of one of the long-distance carriers.

Of course, the call may not be completed. The called party may not answer the phone or the called line may be busy; this information is signaled to the caller by audible tones. These tones, **ringback** and **busy** signals, are distinguishable by the caller because they sound different. Each of the audible signaling tones contains a mix of particular sine waves that compose its spectrum and is presented as an identifiable tone followed by silence. Tones are further differentiated by cadence (the rate at which the tone-silence cycle repeats) and duty cycle (the fraction of each cycle in which sound is present). For example, ringback occurs in six second intervals in the United States with two seconds of sound followed by four seconds of silence.

Other signaling tones convey different information. The **reorder** (or **fast busy**) sound indicates that the network, not the called number, is busy. In the case of a caller error such as dialing an unassigned number or a number that has been changed or not dialing the required “1” for a toll call, a series of three tones in a rise-fall pattern, known as **intercept** (or “special information tones”) is heard, usually followed by a digitized voice explaining the error. All these sounds are indications of **call progress**, i.e., the various steps a call may take towards completion. Call progress tones may be distinguished by differing periods and

⁵ “1” is also the United States’ country code for international dialing so the same number can be used to call across the country or from another country.

duty cycles, and they are designed to be recognized easily by human listeners.⁶ For a computer to recognize the tones it must sample the audio signal from the telephone line to detect cadence, and additionally it may need to analyze the signal's spectrum, which is a more computationally intensive process.

Dialing can also be accomplished by touch tones. When a phone is taken off hook and current flows, depressing a key on the keypad generates a sound, technically called a **DTMF (dual tone multifrequency)** signal. This tone is the sum of two sine waves, one specifying each row and one specifying each column of the keypad. The frequencies were chosen to allow maximal distinction between their harmonics. A decoder in the central office detects the two frequencies and determines which digit was dialed. In addition to the 10 digits, touch tone phones include two extra keys: "*" (asterisk, star) and "#" (sharp, pound sign, number sign, or octothorpe).⁷ Telephone services rely on these keys to differentiate the digits directed to the service from dialed digits specifying a destination number. For example, to cancel variable call forwarding on my telephone I may enter "*73" when I hear the dial tone.⁸

The fact that a telephone has buttons instead of a dial does not necessarily mean it uses DTMF tones to transmit digits. Many less expensive phones have buttons (they are much cheaper to build than dials) but actually use a series of timed make/break pulses like a rotary dial to send the digit. Some of these telephone sets have an option switch between tone and pulse, which is especially useful if a subscriber is not configured for touchtone service at the central office (the local operating company often charges a monthly fee for it) but still wishes to interact with tone-driven interactive telephone services.

The frequencies and amplitude of touch tone signals were chosen to transmit well over the telephone network. When you push a button on your touch tone phone, your phone mutes since the tone generator is quite loud, as someone on another extension will testify. The amplitude makes touch tone detection easy and reliable even over a long-distance connection, which is beneficial for interactive services. It may be difficult to detect DTMF tones while the parties are talking as speech contains energy at a variety of frequencies that interferes with tone detection. Speech may contain frequencies close enough to the touch tone components to trigger false tone detection, especially in less expensive tone detector integrated circuits. This is not a significant issue at the central office, however, as the switch listens only to set up the call, i.e., before any conversation has begun.

⁶There is some variation in the spectrum and cadence of call progress tones. Technical recommendations describe these tones, but network operating companies are not forced to comply. The sounds are consistent enough in the United States that we can usually recognize them but they may be very different abroad.

⁷Actually, the DTMF signaling system supports 16 keys. The extra column, labeled "A" through "D," does not appear on most telephones, however. The extra keys are intended to select a call priority on military telephones.

⁸Of course, I must remember that "73" cancels call forwarding. We will return to the topic of user interfaces to telephone services in the next chapter.

The signaling sounds contain information about the state of the call because they are passed over the same circuit path as the voice signal; they are called **in-band** signaling. Various forms of **out-of-band** signaling may also be provided (e.g., on a key set a light for a particular line will turn on when another extension of that line goes off hook), but out-of-band signaling is mostly limited to digital telephone systems.

When the destination is specified fully by the proper number of dialed digits and the destination line and network are not busy, the destination telephone must audibly alert the called party to the incoming call. This is done by the local switch at the receiving end, which periodically sends a ringing voltage of roughly 100 volts AC down the loop, superimposed on the 48 volts DC. In a traditional phone, this voltage is applied to the windings of an electromagnet; a doorbell-like mechanism then causes a small hammer to vibrate between two bells, producing the classic telephone ringing sound. In more modern phones this signal is detected by a circuit, which makes a sound to imitate the ring using a small transducer. Telephone devices for the deaf detect the ring signal and flash lights for a visible alert. Although this signal as well as the sound of distant ring back that the caller hears are generated in the switch to which the destination phone is connected, the two sounds are often out of phase. This leads to situations where the called party may answer on the first ring, but the calling party has not yet heard any ringback indication.

During the connected portion of the call, the voices of the two parties are transmitted as explained in the next section. Any touch tones generated during this phase of the call are ignored by the switch.⁹ When finished, the parties terminate the call by hanging up their telephones. Going on hook brings the loop back into the high impedance initial state and allows the network to free any resources that may have been associated with the call, i.e., the voice circuit. This termination process is referred to as **tearing down** the call.

Additional in-band signaling may occur during the conversation. For example, a click or special tone indicates an incoming call during a conversation if one party has **call waiting** service. In response, the called party may briefly depress the hook switch (known as a **hook flash**); the switch detects this brief change of line state and puts the first call on hold while completing the second. While on hold, the first call remains connected through the network to the central office associated with the called party, and the circuit resources associated with the call are not freed (nor is billing suspended).

Another new form of signaling information on analog lines is **calling number identification** currently being introduced by regional telephone operating companies. The calling telephone number is transmitted as analog audio in the silent interval after the first ring, using modem-style encoding of digits. Special equipment on the called line can display this number or send it to a local computer.

⁹A few pay phones disable tone generation during the call often to deny callers the option of selecting a long-distance carrier of their choice.

Calling line identification is not yet available on a widespread basis, so the caller's number is generally available only for local calls. Calling number identification is part of a range of new signaling services currently being deployed. These new services include **call blocking** (automatically rejecting calls from a particular number), **distinctive ringing** for calls from a particular number, **call back** (placing a return call to the number that most recently called without the caller knowing what that number was), and **call trace** (which stores the calling number in the switch to be disclosed to law enforcement agencies acting on charges of telephone harassment).

Yet another form of in-band signaling is the **stutter dial tone**, which is associated with voice mail when a message-waiting light is not available. In this situation, the dial tone is not a continuous tone but instead is interrupted, i.e., it stutters, which indicates the presence of new messages in one's voice mailbox. Stutter tone may also be used to indicate an alternate call appearance. For example, an exchange may allow a call to be put on hold by a hook flash. A stutter dial tone then indicates that a new call may be dialed and reminds the caller that there is already a call in progress. After the second call is dialed, the two calls may be conferenced together if desired.

There are many external devices that connect to analog telephone lines, such as answering machines, computer modems, and facsimile machines. These devices employ the same kind of signaling techniques characteristic of analog systems even though some devices transmit data instead of voice. These consumer devices tend to be taken for granted and may be sorely missed in a digital telephone system, which, as described below, uses very different signaling methods.

Transmission

Voice communication can commence once call setup completes. In the simplest case in which both lines are on the same local mechanical switch, the two loops are electrically joined through switching relays and current flows through the completed circuit. Voice is transmitted as current variations in the circuit. The battery from the central office is connected through the local loop to the handset microphone. This microphone typically is full of carbon granules behind a diaphragm located just inside the "mouthpiece." Since sound is composed of variations in air pressure over time, the talker's speech will move the diaphragm, changing the resistance inside the microphone. The loop current changes in proportion to the change in resistance. At the receiving end, the varying loop current flows through a coil around a magnet, which is connected to another diaphragm similar to a conventional loudspeaker. As the diaphragm moves in and out, sound pressure variations are reproduced corresponding to those that impacted the microphone. As a result, we hear the calling party.

The task of the switch and telephone network is to construct a path or circuit for the call. The circuit may actually be a single electrical circuit involving physical connection of the loop coming from the originating phone with the loop terminating in the destination phone. Originally all calls were carried on individual

circuits with patch cords in the central office switchboards completing the circuits. Now it is most likely that once the circuit leaves the local central office, the voice is digitized. The digitized voice is then combined with several other digitized conversations, transmitted some distance by wire, fiber-optic cable, microwaves, or radio transmission to a satellite, eventually demultiplexed from the stream of many conversations, turned back into an analog signal at a distant central office, and sent down the final loop. This entire procedure is virtually transparent to the user except possibly for the delays associated with transmission time on satellite links; we may think of the network as switching virtual analog circuits.

Recall that the loop is a two-wire system. Both parties may talk at the same time and consequently one transmit signal (what one party says) and one receive signal (what the other party says) become mixed on the line. A **hybrid** circuit in the telephone attempts to separate these signals; this is a difficult task in that each connection has a different phase and amplitude response. In other words, the acoustic response of a line varies as a function of frequency, which, in turn, varies from call to call. Some of the signal from the talker's microphone comes back to the talker's earpiece: this is called **sidetone** and provides useful feedback to the caller as a cue to the line quality and a hint as to when to speak up.

But there are several drawbacks to the mix between transmit and receive signals. With telephone-based speech recognition, it is difficult to build an application in which the caller can interrupt ("bargue in") while the computer is talking because the computer's speech will be feeding back into the recognizer. Speaker phones encounter feedback problems when the transmitted signal comes out through the speaker and back into the microphone.¹⁰

Consequently, speaker phones are **half duplex**, meaning that at any moment they are either transmitting or receiving but not both concurrently. Whichever party speaks louder will monopolize the channel ("get the floor"). This makes it much harder to interrupt or offer back channel responses (See chapter 9) and as a result conversation becomes awkward. Several companies (Shure, NEC, VideoTelecom, to name a few) sell full duplex echo-cancellation systems that can be used in a telephone environment. These more sophisticated systems determine the impulse response of the system, i.e., the correspondence between the transmitted and received signal to properly subtract the transmitted element from the received element of the conversation. These devices are still relatively expensive (\$1000 and higher in 1993) pieces of equipment intended for conference rooms, but lower priced consumer models for the desktop are just around the corner.

Telephone audio is band limited to lie between approximately 300 and 3100 Hz. The lower bound is established by blocking capacitors that separate the audio

¹⁰This is not a problem with handsets since the signal coming out of the earpiece is not strong enough to make its way into the mouthpiece. A piece of cotton may often be found inside the handset to prevent acoustic coupling through the hollow body of the handset.

signal from the battery current. The higher frequency is set in part by characteristics of the carbon microphone and in part by the constraints of the digital portions of the network. It is important to limit the upper frequency. The digital segment of the network carries the signal sampled at 8 kHz with 8 bits of μ -law encoded data per sample. As discussed in Chapter 3, μ -law encoding is a logarithmic pulse code modulation scheme, and these 8 bits allow about 12 bits of real dynamic range. The 8 kHz sampling rate cannot represent a signal with frequency components exceeding 4kHz without introducing aliasing distortion; this translates to a little higher than 3kHz for real-world audio filters.

Because of the limited bandwidth some acoustic information is lost during a telephone conversation. A significant amount of low frequency information is lost, generally including the fundamental frequency of voicing, although this does not prevent us from perceiving the pitch of the speech. The higher frequency limit interferes most significantly with the intelligibility of fricatives. These particular frequency limits were chosen with care to allow adequate intelligibility at minimum cost (in terms of bandwidth). Much of the trouble that consumers experience with understanding telephone conversations stems from noisy lines or inadequate amplitude rather than with the bandwidth limitations. Still it is often more difficult to identify a caller over the telephone than in person. On a positive note, users may find synthetic speech less objectionable over the telephone as they are already accustomed to its band-limited characteristics.

Conference calls allow more than two parties to be connected simultaneously. Such a call may be established by a single subscriber with conference service, which allows two calls to be merged by one line; the calls are actually merged in the switch at the request of the common telephone. Local conferencing usually can merge three callers, i.e., two incoming calls; for a larger group a conference service must be involved to establish a **bridge** across the circuits. With a limited number of lines, conferencing can be accomplished by simply adding the signals and perhaps decreasing their amplitude first to prevent clipping speech when multiple parties speak simultaneously. For more than two or three parties, simply adding all the signals is undesirable since the background noise accumulated from each line results in a poor signal-to-noise ratio for the single person speaking. It is better to add together the signals from the one or two lines producing the loudest signals, presumably those conferees who are talking at a given time, and mute the remaining lines to keep their noise out of the conference. This technique interferes with interruption; if many parties speak at once, some will not be heard. Communication is also horribly confounded if inexpensive half-duplex speaker phones are used.

Conference calls are often used as the audio paths for teleconferences in which multiple sites share voice as well as video. Teleconferencing usually requires specially equipped conference rooms. There is growing research interest in less formal conferences in which low cost video equipment is located in ordinary offices, connected together for "video calls," and possibly sharing windows of workstation screens among conferees.

Devices such as modems and facsimile (fax) machines transmit digital data over analog voice circuits. They do so by generating different sounds for "1" bits

and for “0” bits.¹¹ The sounds for various bit patterns are transmitted with differing frequency and/or phase, and the receiver decodes these sounds back into digital data.

DIGITAL TELEPHONES

A digital telephone is one connected to the local switch via a digital connection for both signaling and transmission. Most modern telephone switches are internally digital because digital technologies are often less expensive than analog, digital signaling techniques allow for easier extensions to support new services, and a digital audio signal can be more easily transmitted without degradation.

Unfortunately, all early digital switches used vendor-proprietary signaling techniques, which allowed use of telephones produced by that vendor alone and frustrated attempts to build telephone interfaces for computers. But an international digital standard, ISDN (Integrated Services Digital Network), has been emerging over the past decade, and is the focus of this section. The chief advantage of ISDN over other digital protocols is that it is a standard that ultimately will allow interoperability of equipment across vendors and among countries, making hardware and applications development much more attractive. Unfortunately, true portability across vendors and among countries has not yet been achieved even under ISDN.

With ISDN the connection to each telephone is digital and supports three simultaneous channels called “2B + D.”¹² In such a configuration (Basic Rate ISDN), the 2B, or “bearer,” channels can each carry 64,000 bits of data per second in each direction, which corresponds to the familiar 8 kHz 8 bit μ -law encoding of speech. Speech is usually carried on the B1 channel with the B2 channel reserved for a data call such as the connection of a terminal to a host computer. The 16,000 bits per second D or “data” channel is used for signaling between the phone and the switch. Thus, ISDN signaling is entirely out of band; signaling occurs over a channel entirely different from that which transmits voice.

There are two, three, or four pairs of wires between the switch and the digital telephone depending on how power is supplied to the telephone, but unlike analog phones a pair is not dedicated to each channel. Rather, one pair of wires is for transmit, one is for receive, and the third pair supplies power to the telephone. The bits associated with both B channels and the single D channel are merged together into small data packets (see Figure 10.3), and electrically encoded in a scheme that maintains a balance of “0” and “1” bits on the line.

¹¹Actually most current modem schemes encode pairs of bits, i.e., “00” will be encoded one way and “01” another.

¹²Other configurations are supported. Primary rate ISDN (PRI) supports 23 B channels and a single D channel. Broadband ISDN provides even greater data rates.

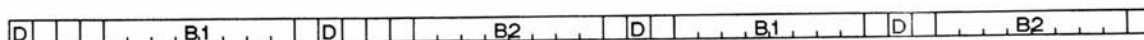


Figure 10.3. The bits of two B channels and one D channel are combined in a single packet.

Signaling

All signaling takes place over the D channel. As with most data communication schemes, ISDN call control may be broken down into a number of layers. The lowest layer, Layer 1, specifies the physical characteristics of the signal including the encoding schemes to balance the bit levels on the wire. Layer 2 defines the addressing scheme, data verification, and retransmission requirements. Layer 3 defines a protocol, Q.931 (ANSI T1.607 in the U.S.), for signaling between the telephone and the switch. Layer 3 is our primary concern for signaling purposes. ISDN currently defines no services above Layer 3. Signaling under ISDN uses data communication schemes similar to those used in computer communication. Although attractive for its power, implementing the ISDN protocol is not trivial, and ISDN telephones (or “terminals” as they are called) or computer interface cards (“network adapters”) are much more expensive than current analog equipment and usually incorporate microprocessors to track the state of each call in progress.

The Layer 2 ISDN D channel protocol is LAP-D (LAP stands for “Link Access Protocol”), or Q.921. LAP-D defines the format of data packets and provides for error checking, retransmission, and flow control. LAP-D also supports a two-tiered addressing scheme to allow multiple data link connections simultaneously over a single D channel. The SAPI (Service Access Point Identifier) portion of the address specifies a service, e.g., a SAPI of 0 identifies the call control service that manages calls on the B channels. Another service, which would use a different SAPI, could be an electronic meter reading service. Digital utility meters could be attached to the ISDN telephone line and accessed from a remote location over the D channel. The second address field is the TEI (Terminal Endpoint Identifier), which corresponds to a particular device being served. Multiple telephone extensions each have a unique TEI so that each can send and receive independent call control messages to the central office switch. For the example meter-reading service, the gas, electric, and water meters might each have its own TEI.

Contained within the LAP-D packets are Layer 3 messages. Layer 3 includes the Q.931 protocol, which implements basic call control. Another portion of Layer 3, Q.932, is designed to implement supplemental services, although the definition of these is still evolving.¹³ Q.931 provides a protocol to set up and tear down voice connections on the B channel as well as data connections on the B or D channels.

¹³Supplementary services include conferencing, call transfer, hold, and drop, as well as managing multiple directory numbers on a single key-set telephone. Most supplementary signaling is at this date switch specific. National ISDN One is an attempt to standardize a core of supplementary services in the U.S.

When a telephone is first plugged in, it brings up its Q.921 layer and establishes a Q.931 link with the switch after negotiating a TEI. When a calling party picks up the handset, a SETUP message is sent to the switch to establish basic parameters such as the type of call; the switch responds with a SETUP ACKnowledgment (see Figure 10.4). Individually dialed digits are then sent as a series of INFO messages to the switch. If all the digits are known in advance, they can be sent within the SETUP message. When enough digits are dialed, the switch returns a CALL PROCEEDING message. Most current ISDN telephones generate audible touch tone frequency sounds during dialing; these tones provide both feedback to the caller and once the call is established in-band signaling support for nonISDN interactive services. During dialing, there may be no B channel established to the switch and the switch does not listen to these tones.

The telephone of the called party then receives its own SETUP message, which may contain INFOrmation fields identifying the number of the calling line as well as other information. The telephone set may use this information to post a message in its display if it has one. The phone itself then generates a ringing sound; this is produced locally not by the central office (although the central office may specify *which* of several different ring patterns to use in support of services requiring distinctive ring sounds). The phone sends back an ALERTing message

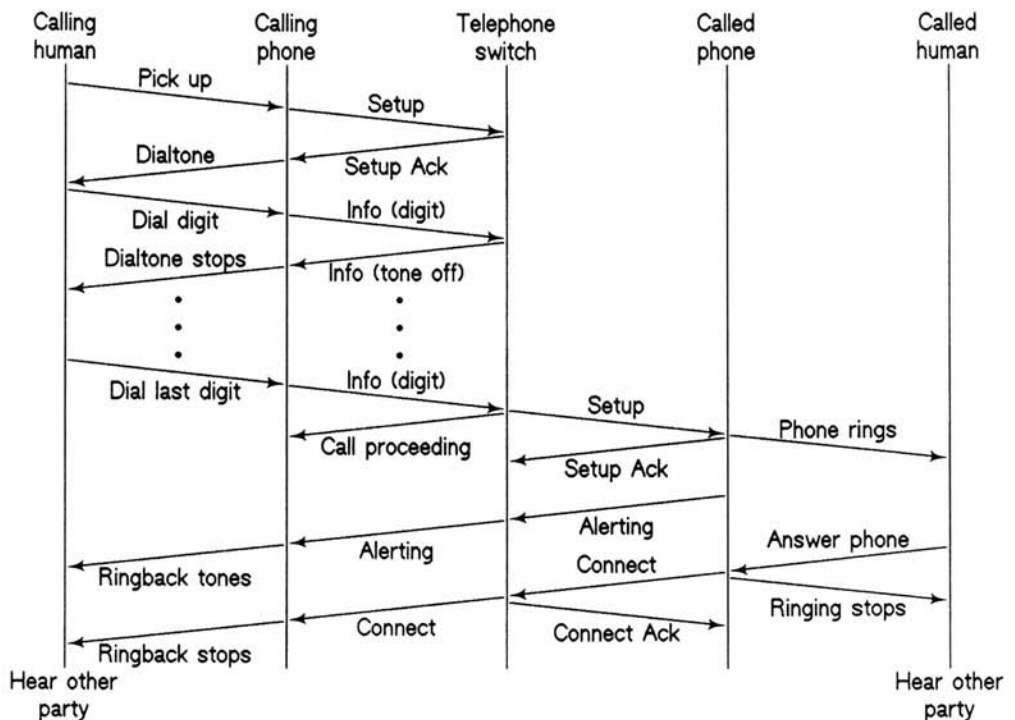


Figure 10.4. The exchange of messages in setting up an ISDN voice call.

to the switch, which passes an ALERTing message back to the calling telephone, which can then generate a ringback indication. If the called party answers the phone, a CONNECT message is sent back to the switch. The switch acknowledges this CONNECT message to the answering party and passes it on to the calling party. At this time, the B channels for each phone are connected and digitized audio is exchanged. At the end of the call, one party hangs up the telephone, which generates a DISCONNECT message, and the switch passes on a RELEASE message to the other phone. Thus, all parties (calling phone, called phone, and both switches) know that the call has been torn down.

Each Layer 3 message is a packet containing a number of fields called information elements. Some information elements are mandatory, some are optional, and, at this point in time, many are specific to a particular switch vendor; these elements are implemented using special “code sets” for each switch. Figure 10.5 shows a sample message and its contents.

Transmission

Digitized voice is carried easily on a B channel. A codec in the telephone digitizes the signal from the microphone as 8 bit μ -law encoded samples at 8 kHz and transmits the samples on the appropriate B channel. At the receiving end, another codec turns the digitally encoded speech into an analog signal, which then drives the earpiece. Applications such as digital answering machines can capture the digital voice data directly from the network to store in a file. An ISDN network is all digital, which allows end-to-end transmission of the digital signal. The main advantage to the consumer is higher quality telephone connections with no difference in audio quality between local and long-distance calls as no additional acoustic information is lost after the initial digitization. Another feature of ISDN voice transmission is that the two sides of a call are carried independently without the mixing that occurs in the analog local loop circuit.

If appropriate equipment is available at each end of the connection, enhanced audio encoding can be carried over the 64,000 bit-per-second B channel. A technique to apply ADPCM coding to several audio frequency sub-bands gives an effective bandwidth of over 7kHz [Mermelstein 1988].¹⁴ The higher bandwidth is especially beneficial for audio conference calls as it facilitates identifying the person talking.

During call setup under ISDN, pressing keypad buttons on the telephone sends digital messages to the switch; any acoustic feedback of standard touch tone sounds is for the convenience of the person dialing. But during the transmission phase of the call, pressing buttons does *not* send messages to the remote switch or telephone. To allow users of interactive voice response services access to those services over ISDN telephones, audible touch tones must be sent in-band over the voice channel. Some digital telephone systems generate the appropriate touch

¹⁴CCITT standard G.722.

C SAPI= 0 (CCP) TEI=127
 PD=08 UCC Ref=O 10

M 05 SETUP

I 04	Bearer Capability	Len=3
80	Coding standard	CCITT
	Transfer capability	Speech
90	Transfer mode	Circuit
	Transfer rate	64 kbits/s
A2	Layer 1 protocol	G.711 u-law
I 18	Channel Identification	Len=1
88	Interface identifier explicit	No
	Interface type	Basic
	Indicated channel exclusive	Yes
	Channel is D-channel	No
	Channel selection	None
I 1E	Progress indicator	Len=2
82	Location	Local public
83	Progress description	Orig non-ISDN
I 34	Signal	Len=1
4F	Alerting off	

C SAPI= 0 (CCP) TEI= 64

PD=08 UCC Ref=D 10

M 01 ALERTing

I 18	Channel Identification	Len=1
81	Interface identifier explicit	No
	Interface type	Basic
	Indicated channel exclusive	No
	Channel is D-channel	No
	Channel selection	B1

Figure 10.5. The information elements contained in a SETUP message and the ALERTing message sent in response.

tone sounds in the local switch instead of the telephone when the caller uses the keypad, but this provides less effective auditory feedback to the user.

PBXs

Not all telephones are connected directly to the central office. In business environments with a large number of telephone lines, it is more common that the phone in an office is connected to a switch owned by the business called a PBX (Private Branch eXchange). The PBX provides an initial level of switching internal to the organization. As illustrated in Figure 10.6, a PBX can either connect two internal phones to each other (without incurring a charge from the telephone operating company as none of its resources are used) or connect an internal telephone to an external number through trunks to the central office. In an alternative arrangement sold under the trademark Centrex, each telephone is connected to the central office, but software in the central office switch emulates the internal number dialing and other features of an independent PBX.

PBXs provided many calling features, such as conference, transfer, and hold long before these were available to single line customers.¹⁵ Because PBXs usually serve many fewer lines than a central office, they represent a much smaller capital investment and are marketed by many vendors. Because their interconnection with the public switched telephone network is limited and well specified, the PBX is free to implement its own internal call management protocol. Presently PBX vendors offer a variety of incompatible and proprietary digital protocols to support services such as display of calling party and call routing information such as “forwarded from” that can be used by a receptionist answering many lines.

PBXs also offer the flexibility of providing their own numbering plan. For example, in the sample PBX in Figure 10.6 all the internal extensions consist of two digits beginning with “2.” This numbering plan interferes with dialing the outside number 232-1234 as the internal call to extension “23” would be initiated when the second digit of that number was dialed. Although the PBX could delay after the second digit waiting to see if the caller dials any additional digits, this would slow down all internal calls. Instead, most PBXs employ a numbering plan specifying that the prefix “9” precede all outside calls.

PBXs are often used to form private telephone networks within large companies that are geographically dispersed. For example, the PBX we have been using as an example might provide calling to a remote office when the first dialed digit is “8”; to reach telephone “25” at the remote site, the caller would dial “825.” The PBX would route this call to a trunk (see Figure 10.7) leased from one of the long-distance operating companies tied to the distant PBX. The distant PBX would then complete the circuit to the local telephone line numbered “25.” This archi-

¹⁵A set of central office services called CLASS (Custom Local Area Signaling Services) provide these as well as many of the other features discussed in this chapter, but CLASS is only now becoming widely available in the U.S.

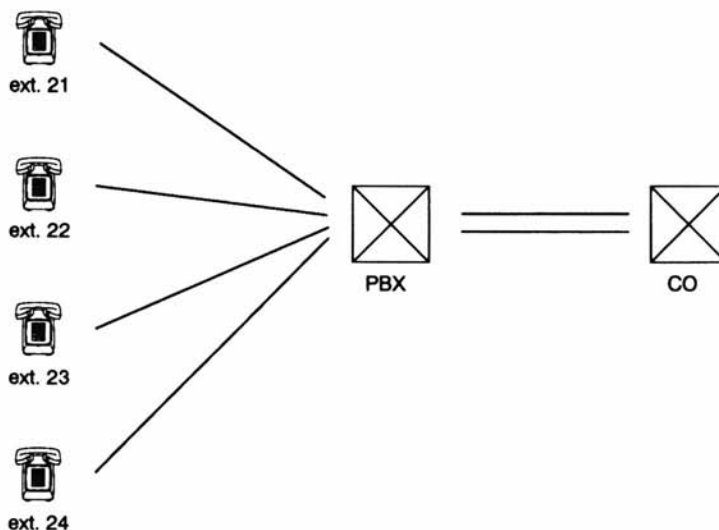


Figure 10.6 A PBX provides an internal telephone network. Telephones served directly by the PBX can be connected without using the public telephone network. The PBX can connect an internal telephone to the central office switch through trunk lines.

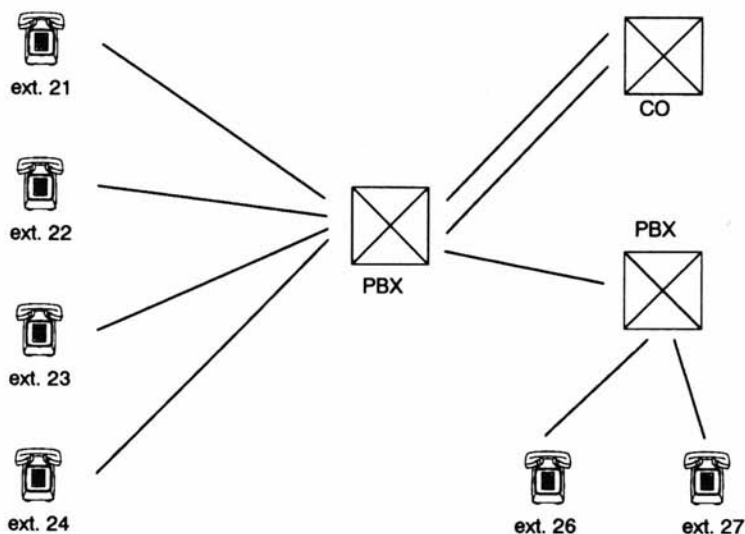


Figure 10.7. A PBX may have trunks to other PBXs as well as the central office, providing private telephone networks.

ture mirrors the arrangement of central offices and connections to remote central offices through the public telephone network; the only difference is who owns the equipment.

In practice it is more common that each telephone number is unique. The PBX would know from its routing tables that telephone numbers in the range of "25" to "30" are at the remote site and route these calls to the distant switch. In any case, the digits "25" must be transmitted to the remote switch so that it can route the call to the appropriate remote telephone.

SUMMARY

The telephone plays an important role in many aspects of using computers for communication. Telephone-based interactive voice services provide remote access to a variety of databases. The next chapter discusses computers as mediators of telephone conversations between people. Chapter 12 describes stored voice as a computer data type; at the moment, most such data arrives over the telephone, e.g., voice mail messages. This chapter has discussed operational characteristics of telephone networks so that the reader appreciates both the underlying issues in using computers to control calls and how properties of the telephone voice channel require particular interaction techniques for voice services.

Telephone management can be broken down into a signaling phase that sets up and tears down calls and a transmission phase during which two or more parties converse. Signaling and transmission are handled very differently in analog and digital networks, although most *functionality* can be had using either. Analog signaling is in-band, i.e., carried as audible sounds. A computer interfaced to the analog network must listen in and analyze the acoustic properties of sounds on the local loop to track the progress of a call or use hardware that does essentially the same thing. Digital signaling uses protocols similar to those used in existing computer networks and hence can be more easily integrated with other workstation operating system activity than analog signaling. Digital signaling is also faster and allows more readily for management of multiple calls simultaneously.

Analog and basic digital voice transmission support similar audio coding quality and bandwidth; the data rate of the ISDN B channel was chosen so as to model the characteristics of analog telephone circuits. But the digital network offers several additional advantages. Foremost in an all-digital network there are no further reductions in audio quality after digitization no matter how many miles the circuit spans; analog circuits exhibit signal degradation at each stage of amplification, which can be numerous over a very long distance. An end-to-end ISDN circuit keeps both parties' voices separate, making it easier to use speech recognition to interrupt an interactive service while it is speaking. The digital network can be used to carry speech based on improved coders offering higher audio bandwidth over existing channel capacity. As ISDN offers multiple B channels on a single telephone line, voice and data call or multiple voice calls may be in progress simultaneously over a single connection to the network.

Although ISDN is attractive for many reasons, it will be slow to penetrate the consumer market because digital telephone equipment is more expensive than analog, and few residential customers require telecommunication services that can benefit from ISDN's capabilities. At least in North America, the first wave of ISDN deployment has been limited generally to select business environments. Although the economics of wide-area networking or the need to merge voice and data services dominate deployment decisions, business environments also have the most to gain from integration of computers and telephones for call management—which brings us to the topic of the next chapter.

FURTHER READING

There are a limited number of references that explain telecommunications technologies to a broad audience. Chorafas presents a survey of network technologies and weaves them into their roles in an evolving global network. Stallings presents ISDN in a very readable manner. Briley focuses on details of switching in the telephone network, while Talley concentrates on data communication protocols; both books are readable by a general audience but more focused than was this chapter.

Telephones and Computers

The previous chapter discussed interactions between people, telephones, the central office switch, and the remainder of the telephone network. In this chapter we consider the benefits computers offer for call management through the enhanced services and flexible call routing they can provide. Computers can provide a user interface more powerful than the keypad or low-power display of conventional telephones. Computers can personalize telecommunications and change the way we think about using telephones.

The signaling and transmission functions of the telephone were explained in the previous chapter. Extending signaling capabilities to computers allows the computer to place calls, employ a variety of means to alert the user to an incoming call, and automatically route calls on the basis of who is calling, the time of day, and the current activity of the called party. Interfacing computers to the transmission phase of the telephone network allows the user to place calls by name or from other personal information management applications. Computers can be sophisticated answering machines providing store-and-forward capabilities so that we can “converse” asynchronously; the telephone may prove to be the primary source of voice as data to the office workstation.

Current digital telephone switches and voice mail systems are in fact computers, some running operating systems similar to those on our workstations. But the programs and run-time environments of these switches are specialized for telephone tasks, such as switching voice circuits in real time, maintaining billing records, and periodically running diagnostic checks on the integrity of the network and its constituent circuits. Telephone switches are designed to execute a specific task quickly and reliably; they are not designed to run user applications.

This chapter examines telephony applications that run on the same workstations we use to edit documents, develop software, or read our electronic mail.

This chapter begins by exploring the benefits of involving a workstation in the call management process and then offers examples of the kinds of services it can provide. The hardware requirements and underlying architectures that allow workstations to participate in call setup are then discussed. Finally, several case studies are presented for a more detailed description of the services that can be offered by interfacing computers to telephones.

MOTIVATION

In many contemporary office environments it is common to find a workstation or personal computer and a telephone in each office, a communal fax machine, and a voice mail system housed in a telecommunications closet. The voice mail system interacts with the telephone via a message-waiting light but otherwise these devices each function in isolation. This chapter proposes linking the functionality of these various devices with particular emphasis on including telephony functions in the computer, from which a number of benefits may be realized.

Access to Multiple Communication Channels

Providing *ease of access* to communication facilities over different media motivates computer-mediated telephony. Facsimile use has exploded, fostering boards that allow personal computers to send and receive faxes as well as display bit-mapped images on the screen. Many users keep their phone number databases on computers and consult them when placing a call, or dial by holding the handset next to their palm-top computer. Electronic mail use is taking off and is now widely available; its existence relies on the computer network, but messages are also being forwarded to pagers and cellular telephone-equipped mobile computers.

Making all these communication media available from a single desktop computer allows the user to easily cross media boundaries during a single communications session. For example, receipt of an email message might imply that the difficult-to-reach sender is in the office and hence available by phone for a more interactive discussion. Then during the course of discussion, conversants could exchange pertinent documents by email or fax. A caller may prefer to send a message by email instead of leaving one with a receptionist if the called party is unavailable.

Integrating disparate communication media at the workstation affords immediacy and convenience to employing multiple media simultaneously or sequentially. Bringing incoming messages into a single computing entity simplifies the task of discovering whether any new messages have arrived and enables a variety of alerting, filtering, or message forwarding policies under control of the recipient.

Improved User Interfaces

A second reason to integrate telephony into the workstation is the potential for significantly more powerful user interfaces than can be provided with telephone equipment alone. Most sophisticated telephones provide a touchtone keypad and perhaps a few dedicated function buttons such as *hold*, *conference*, and *transfer*. User interfaces to all telephony services must be implemented using these buttons, and feedback to the user is limited to a small repertoire of sounds such as the busy or dial tone signals. Without a display, it is difficult for the user to determine the current state of any features that may have been activated. For example, my telephone has speed dialing but it cannot tell me the contents of my speed-dialing list; instead, I have to place the call and see who answers or tape a piece of paper with my speed-dialing list to the phone. Likewise, I can touch a *redial* key to dial the most recently called number but cannot find out what that number is without trying it. I can key “*73” to cancel my call forwarding, but I cannot find out the number to which my calls are currently being forwarded.

Each year a greater number of sophisticated telephone services are offered: call waiting, call back the number that last phoned, block calls from certain calling numbers, assign distinctive ring to some calling numbers, identify multiple incoming lines with a distinctive ring, speed dialing, calling line identification, and voice mail, to name just a few. But the telephone keypad provides a sparse user interface, making it difficult to use even the existing services much less new ones. A few users may read the manual to safely reconfigure their phone, but many avoid or underutilize the new services.

One approach to providing improved user interfaces has been to build more complex telephones, with displays, soft keys, menus, and even voice prompts. However, workstations are already suited for this task. Computer displays are typically high-resolution CRT screens, whereas telephone displays if available are small and difficult to read due to hardware cost and electric power limitations. Workstations offer additional and more elaborate input devices including full keyboards and mice. Even more beneficial is the fact that powerful window systems provide rich and flexible user interfaces and allow several concurrent applications to share the screen. Workstation operating systems allow resources such as memory or digital signal processors to be shared among a number of applications; the hardware in a “smart telephone” is wasted when the telephone is idle.

There is limited experimental evidence on the value of screen-based interfaces to telephony. [Roberts and Engelbeck 1989] compared the computer and advanced telephone user interface media using standard touchtone telephones, display-based telephone terminals, and telephones augmented with speech synthesis for voice menus. They found the display-based interface both faster and most well received by subjects who routed calls, screened calls, and retrieved messages. But long-term studies based on MICE users (Bellcore’s telephony development environment, described shortly) were somewhat less convincing. [Root and Koster 1986] substituted mnemonic key codes for the conventional numeric ones (e.g., “CF” instead of “72” for call forwarding). They found that subjects recalled more

system functions with the mnemonic codes but did not increase their use of the services. Similarly, [Root and Chow 1987] found that users readily employed a screen-based interface to manipulate and edit their speed-dialing lists, but this did not increase the number of entries they kept in the lists. Although subjects could speed-dial by the screen in addition to the keypad, they did not do so. Interestingly, speed-dial usage increased somewhat during the course of the study.

It is difficult to draw strong conclusions from this evidence. The evidence does indicate, however, that while some advantage can be rightfully claimed for improved interface technology, the underlying service offerings may themselves lead to stable usage patterns. These experiments tested limited computer telephony services, and although the "improved" interfaces employed new technology, the new services offered small improvements over those to which they were compared. More sophisticated services can be built from the interactions between multiple functions, e.g., returning the call of a voice mail message with a single mouse click or automatically rerouting calls when the user's calendar indicates that the user is out of town. But it is much easier to measure the effects of a small, constrained change to a user-interface than it is to compare the utility of one set of features with another very different set over an extended period of use. Of the research projects described later in this chapter only MICE included an evaluation component.

Enhanced Functionality

A workstation can offer enhanced functionality by personalizing call setup for both outgoing and incoming calls. Placing a call can be facilitated by allowing the user to spell the name of the desired party with the computer keyboard, speak the name to a speech recognizer, or use another application such as a personal address book to specify the number. Or using interactive cut-and-paste or drag-and-drop paradigms, a user may be able to deposit an email message into a telephone icon to place a call to the mail sender. For incoming calls, the computer may route the call to a different phone according to a predetermined set of conditions or play a caller-specific outgoing voice mail message.

Call routing is the most sophisticated of these functions. Factors influencing how a call might be handled include the identity of the calling party, the time of day, the location and activity of the called party, and whether a receptionist is available to take an unanswered call. For example, an "important" call could invoke a distinctive ring sound or be forwarded to the nearest telephone if the recipient of the call is in another office. A call may be classified as "important" by virtue of having originated from a recently dialed phone number so as to help prevent "phone tag" in which repeated calls back and forth go unanswered. A variety of technologies can be used to determine the location of the called party or deliver notification of an incoming call while absent from the office.

Much of this potential routing information is dynamic and relies on knowledge about the user. When two parties play "phone tag" they alternate calling each other and leaving messages. If outgoing calls are placed using the same worksta-

tion that screens incoming calls, phone tag can be detected, which might initiate a more thorough search, or page, by the workstation in response to an additional incoming call. Similarly, if a meeting with someone is scheduled for tomorrow morning and that person calls today, it is likely to be relevant to the meeting and hence a more important call than otherwise, but this can be known only by accessing the called party's calendar and telephone directory to make the association.

It would be difficult to implement this degree of personalization and style of experimentation with the standard services in the telephone switch.¹ Telephone switches are designed to establish calls according to relatively static routing tables, which are modified by very limited touchtone based user interfaces. They do not have access to personal information such as a user's calendar. An entirely new dimension of flexible call establishment requires more varied per-call conditional routing factors as well as a more sophisticated user interface to manage the more complex routing rules.

Voice and Computer Access

Another motive for connecting telephones and computers is the potential for *remote voice access* to personal databases. Although a personal workstation is capable of storing many megabytes of information, its utility is limited if the data is inaccessible when the user is away from the office. Databases, such as calendars, electronic mail, and telephone numbers can be made available through interactive telephone interfaces using the techniques described in Chapter 6. Once connected to an application by voice, it is useful to be able to add to a database, e.g., to put an entry in one's calendar by speaking. Such voice interfaces may make the telephone the primary source of voice as a data type on multimedia workstations; examples will be offered later in this chapter as well as in Chapter 12.

PROJECTS IN INTEGRATED TELEPHONY

Several research projects have attempted to integrate computers and telephones in different ways. This section describes the set of telephone features of each and the system architectures upon which they were implemented. These descriptions highlight the many benefits of flexible call management by computers.

Etherphone

One of the seminal attempts to merge computers and telephones was the Etherphone project at the Xerox Palo Alto Research Center (PARC). Etherphone

¹It should be noted that in some ways MICE, described later in this chapter, developed an architecture allowing a greater degree of user access to the configuration tables in the telephone switch. This certainly facilitates call setup services, but such tables alone cannot deal with dynamic call routing depending in real time on user input or situational information.

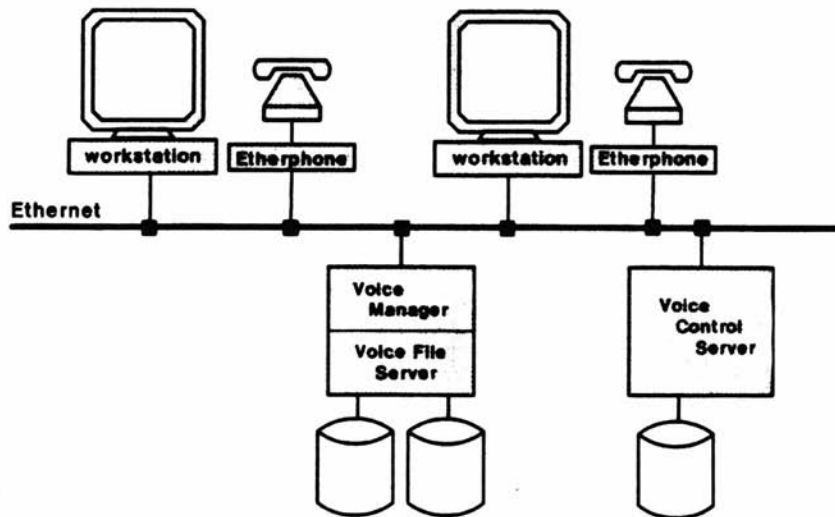


Figure 11.1. Etherphone architecture. (From Terry and Swinehart, "Managing Stored Voice in the Etherphone System." Reprinted with permission from *Transactions on Computer Systems* ©1988, ACM.)

included hardware to digitize voice and control analog telephone lines, both under the control of a workstation in concert with various networked servers. The project included call control applications, integration of audio with the local computing environment and its associated user interfaces, and voice-oriented applications such as audio editing, voice annotation of text, and scripted documents.²

The Etherphone itself was a hardware device, capable of digitizing and playing back speech over an Ethernet. Its physical components included a speaker, microphone, and a telephone handset, as well as a standard analog telephone line interface, which enabled calls to be placed over the external telephone network. Internal "calls" were transmitted over the Ethernet. During call setup, a centralized call control server negotiated with the user's workstation and then sent commands to an Etherphone agent, which dealt with the physical devices in the Etherphone and the stream of audio data packets. Voice was transmitted over the Ethernet either between Etherphones or between an Etherphone and a variety of servers as shown in Figure 11.1. The servers included a storage server to save or retrieve sound from disk, and a text-to-speech server that received text and transmitted the synthesized speech over the network.

Internal and external call processing proceeded differently within the Etherphone environment. In the case of an internal call between Etherphones, the entire conversation was transmitted over the Ethernet and local distributed pro-

²See previous discussion of scripted documents in Chapter 4.

cesses coordinated the setup and tear down of the call. For external calls, the Etherphone provided ring detection for incoming calls and touchtone generation to place a call, but switching occurred outside Etherphone in the ordinary telephone network.

For basic telephony, Etherphone provided several methods of dialing a number including a personal directory and dialing by name. A log was kept of all call activity. Tools, primarily text-based (see Figure 11.2), provided user interfaces to the dialing functions. For incoming calls, each Etherphone user had a distinctive ring "tune" to distinguish whose phone was ringing when within earshot of several offices. If the caller was also an Etherphone user, his or her ring tune alternated with that of the called party, giving an audible form of calling party identification. A telephone icon on the screen also indicated the incoming call and visually displayed the caller's identity.

With local call routing and voice distribution over the Ethernet, a number of new services were possible. The tight binding between a name and a telephone number could be broken; while the called party was logged in on any workstation, Etherphone calls could be forwarded to the nearest Etherphone. While in another office, an Etherphone user could enable "call visiting" from any workstation, causing one's calls to forward to that office. Because incoming calls were indicated with the distinctive ring tune, the host and visitor knew who should answer the telephone. A meeting service allowed multiple listeners to tune in to a meeting

Etherphone											
Phone	Answer	Disconnect	SpeakText	StopSpeech	Directory	Drop Out					
Called Party: Aquarius Theater info				Calling Party: wyatt.pa							
40: Speaking text: "Suppose Alexander Graham Bell had waited..."											
November 12, 1986 11:45:52 am PST											
52: Placing call to Aquarius Theater info (327-3240)											
Call to swinehart.pa at November 12, 1986 11:24:18 am PST is completed, duration = 00:01:08											
Call to Recording service at November 12, 1986 11:26:46 am PST is completed, duration = 00:00:55											
Call from outside line at November 12, 1986 11:33:17 am PST was abandoned, duration = 00:00:06											
Call from wyatt.pa at November 12, 1986 11:35:08 am PST is completed, duration = 00:01:04											
Call to Text-to-Speech service at November 12, 1986 11:42:40 am PST is completed, duration = 00:00:39											
Call to Aquarius Theater info (327-3240) at November 12, 1986 11:45:53 am PST is in progress											
Telephone Directory [0] polled 00:00											
Clear	Reset	Get	GetImpl	PrevFile	Store	Save	Time	Split	Places	Levels	(C) Log
Name				Office		Home		Details			
Services											
AAA Emergency Service				595-3411		408/246-5811		Palo Alto, Mtn View			
Time Announcement				767-2676		767-2676					
Aquarius Theater info				327-3240							
PA Square Theater info				493-1160							
Enrico's Foreign Car				961-4848		*		Fiat repairs, 2145 O. Mdfd MV			
Dr. Kanemoto, Benson				326-6319		*		Dentist			
Dr. Stegman, Deidre				321-4121		*		TakeCare Primary Care			
physician											
Menlo-Atherton Insurance				329-1150		*		renter's insurance: Eleanor			
A Time For You				941-7034		967-9180		haircuts			

or lecture from their separate offices; remote participants could speak to the meeting as well as listen in. The meeting service was implemented by multicasting the voice packets from the meeting room Etherphone to those of the parties listening in.³

Etherphone implemented fully-distributed call control by allowing workstations to place calls through requests to the connection server, i.e., a "software PBX." Both voice and call control data passed through the local area computer network. Calls originating outside the Etherphone service area were also handled in a distributed manner as each workstation's Etherphone interfaced directly to the outside analog lines.

Through its layered architecture, Etherphone brought a greater degree of sophistication to call management. The flexibility of its hardware and network implementation enabled broadened concepts of a telephone "connection." As a highly integrated environment, Etherphone made major contributions for voice applications development in the context of a larger operating system; such styles of integration are the focus of Chapter 12. It was known for its variety of applications for voice in documents as discussed in Chapter 4. Finally, Etherphone helped extend the concept of *services* by partially bridging the gap between the services offered by the telephone industry such as call forwarding and those offered by the computer industry, such as file storage.

MICE

MICE (Modular Integrated Communications Environment) at Bellcore (Bell Communications Research) was built as a testbed for prototyping enhanced services and evaluating their use over time. The subjects of these evaluations were local laboratory members, who used MICE on a daily basis from 1985 to 1988. Although many of the trial services implemented under MICE were fairly simple, MICE provided a foundation for forging closer links between the workstations and telephones in each person's office. The trial services included "memory dialing," a mnemonic form of speed dialing, and MICEmail, an integrated voice and text message system. MICEmail provided both local and remote (using speech synthesis) access to both types of messages.

MICE explored several issues from the point of view of a local telephone operating company by defining the services and resources to be accessed over a future telephone network. MICE placed emphasis on *personalization*; messages were sent to users instead of terminals with the network turning the name into an address and determining a suitable delivery mode. These services were designed to be *customizable*; the user-configured services by changing a profile table within the MICE system. Finally, MICE worked with *integration* of voice and text message types.

³Broadcast sends the same packet to every host on the network. Multicast also sends a single packet, but only selected hosts read it.

The MICE architecture was based on a centralized server, which provided basic telephony functions as well as voice storage, speech recognition, and text-to-speech synthesis. Telephony functions were provided by a small PBX closely controlled by a workstation, which acted as a server to other computers on the local area network. A central control process on the server controlled the telephone switch by executing a finite state machine defined by a call control table, which was accessible to other workstations on the network. The call control tables could be programmed to implement the new server by means of a graphical “service editor,” but making changes to the tables was a complicated procedure. Some work was done to explore screen-based interfaces to such simple service definitions such as time-of-day dependent call forwarding.

MICE was considered to be a prototype open-architecture central office telephone switch in which switching is performed entirely within the switch but outside entities can program how switching occurs. Although its primary purpose was to provide a flexible service development environment for its creators, MICE

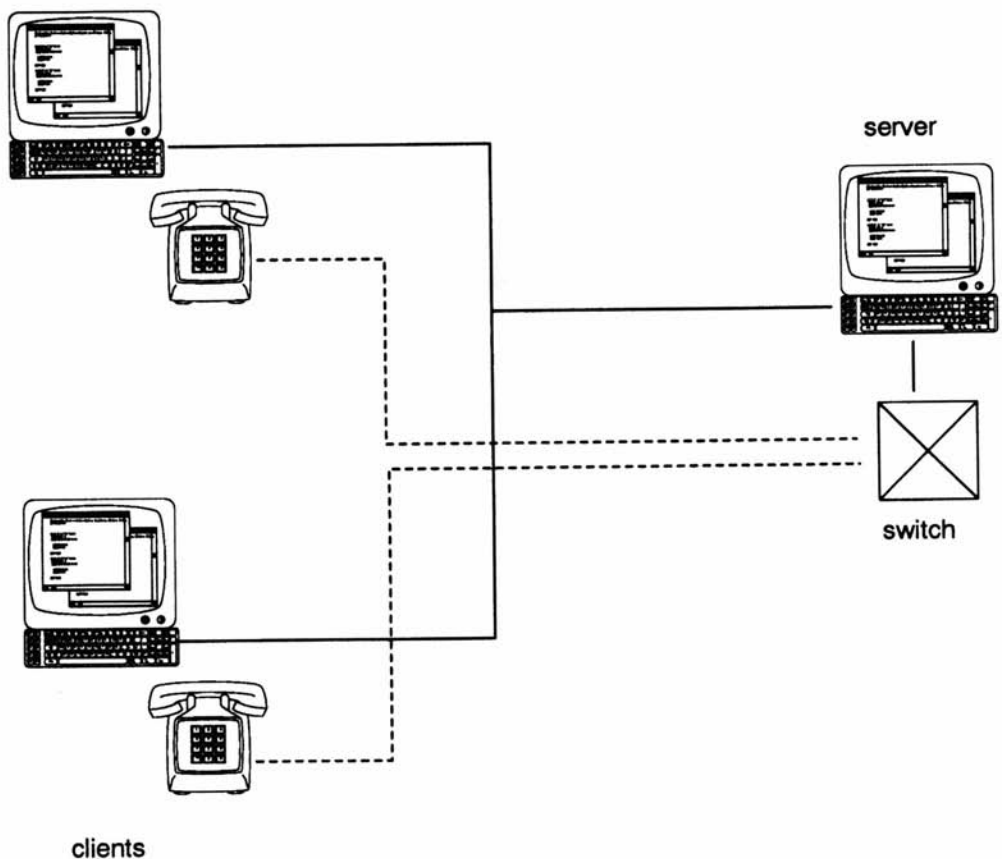


Figure 11.3. MICE architecture. Desktop computers interface to telephones via a server computer controlling a telephone switch.

also allowed computer access to the call control tables that described how calls were handled. Client workstations did not handle each call as it came in; rather, they could update the tables which the server used to route subsequent calls.

Being centralized, MICE transmitted audio to users' offices for services such as voice mail over the telephone network. Even when using a screen-based interface to access voice messages, a user was required to pick up the telephone to hear a message. Using telephone wiring to transmit audio from a specialized server was advantageous at a time when each workstation could not play audio locally. However, the inconvenience limited voice to a few applications instead of allowing it to become a widely available data type to multiple applications.

BerBell

BerBell,⁴ also from Bellcore, was used for telephone and voice services for a real user population, and it offered increased flexibility for the creation of new services under control of the users' workstations. The Berbell project was initiated in 1985, became operable in 1986, and continues in use today. Some of the more well-known services provided by Berbell include speech synthesis for a national weather forecast, a music synthesis demonstration, and local voice paging using speech synthesis over a public address system.

Berbell utilized three servers: one for the telephone switch, one for speech synthesis, and another for digitization and playback of audio. The system included a number of basic services and features that could be invoked by the user with touch tones; in this sense, it was not very different from many PBXs. But since the Berbell servers were also on the local computer network, the same features could be activated from the user's workstation. Additionally, a programmatic interface allowed users to program new services. Over its lifetime, Berbell evolved in the direction of providing complete control over telephone calls at the workstation; in this sense, it allowed much more dynamic call control than MICE.

Personal eXchange

The Personal eXchange (PX) project at the Computer Research Laboratory of Bell Northern Research took a distributed approach to the integration of voice and telephony with computer workstations [Kamel, Emami and Eckert 1990, Bowles *et al.* 1991]. PX also employed a central digital switching facility (a Northern Telecom Norstar PBX); however, this was an "open" PBX, and all routing commands were delivered by computers and telephones distributed around an office. As shown in Figure 11.4, located on each desk were the following.

- A telephone used for its microphone, speaker, and to serve as backup equipment if the computer failed.

⁴Named after the login ID of its creator, Brian E. Redman.

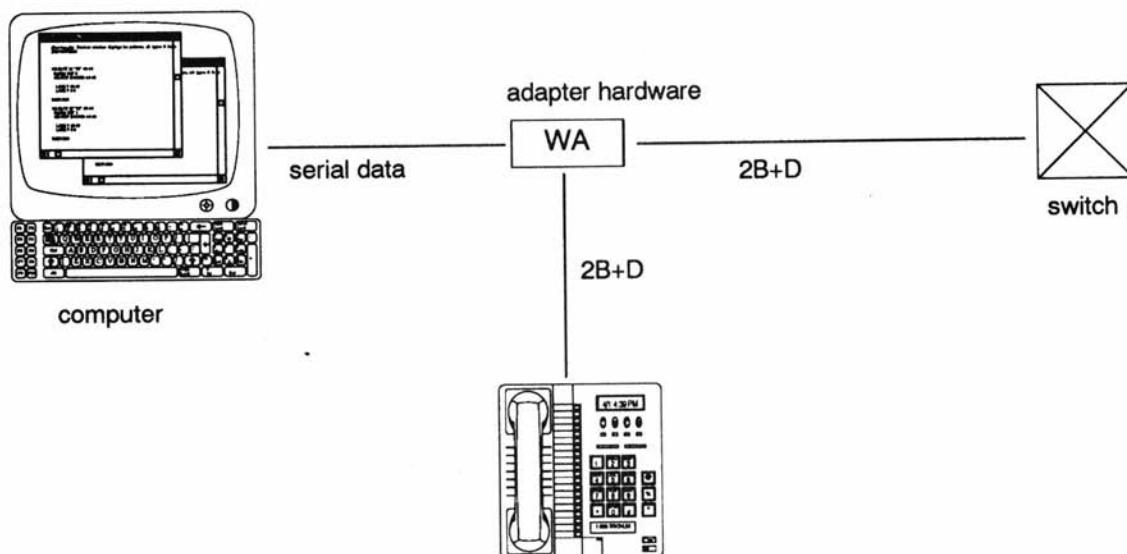


Figure 11.4. Architecture of the PX project. A workstation adapter, WA, allows a computer to interface to both the telephone instrument and the telephone switch by converting digital signalling protocols.

- A “workstation adapter” that brought both speech and signaling information to the computer via a digital telephone line.
- A computer, that controlled the telephone set in addition to call processing.

The wiring of the digital telephone system was used for both voice transmission and exchange of messages between call management applications and the telephone switch. Protocols similar to ISDN allowed simultaneous transmission of voice and signaling messages. When a call arrived, the switch broadcast a call setup message to all workstations with access to the dialed number. Multiple recipients could respond; for example, a screen-based telephone (such as that shown in Figure 11.5) or the telephone set next to the workstation could try to claim the call. Whichever recipient responded to the call sent an acceptance message to the switch and a voice connection was established.

A wide range of applications were implemented around this architecture; the telephone dialing interface of Figure 11.5 is but one example. In another PX application, a conversational answering machine took an interactive message by playing prompts and recording answers, and a graphical user interface allowed message retrieval (see Figure 11.6). Towards the end of the project, speech recognition and speaker verification were being incorporated into the workstation as part of the set of local services available to applications. Although much of PX focused on providing mechanisms for distributed call management, it was also concerned with the larger issues associated with the management of voice in the workstation (such as the voice editor described in Chapter 4) along with software toolkits to support audio and telephony in applications.

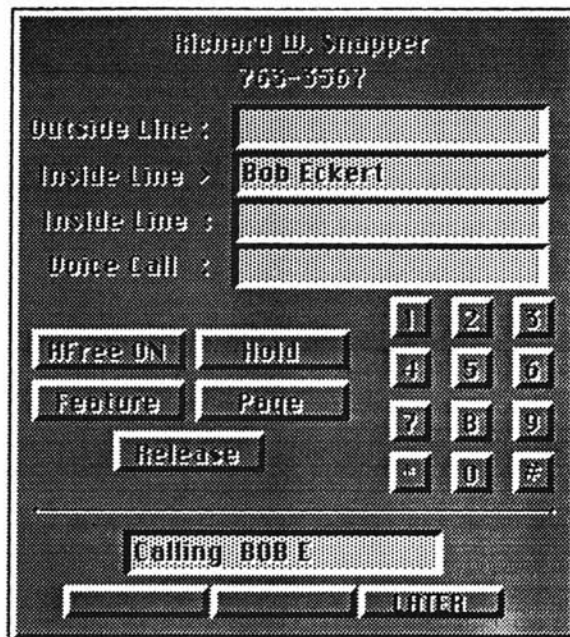


Figure 11.5. A PX screen-based telephone application. (Printed by permission of BNR.)

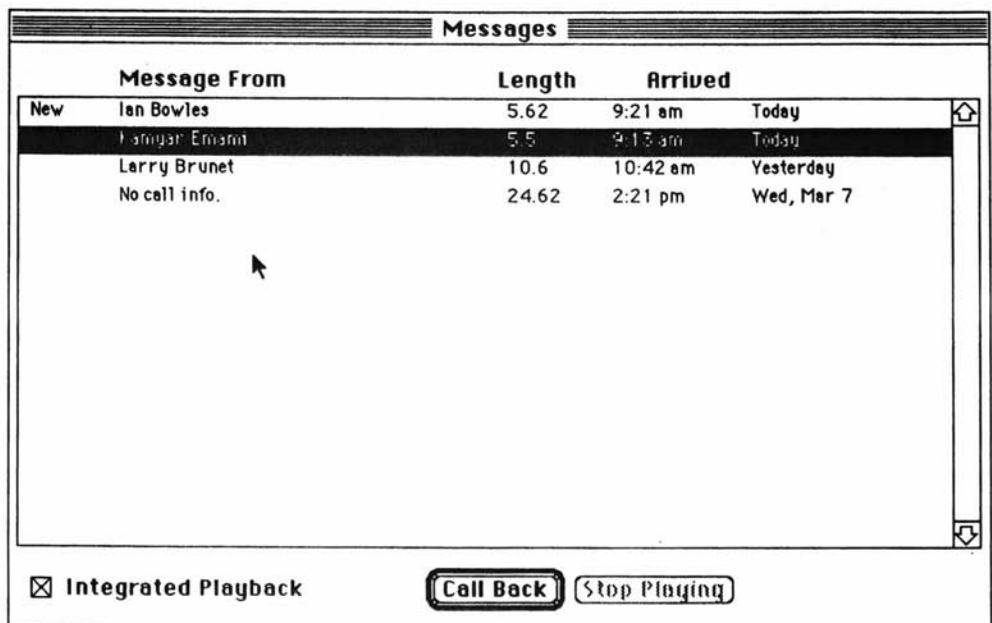


Figure 11.6. The display of a PX answering machine program. (Printed by permission of BNR.)

Except for the PBX, PX was a completely distributed environment. The connection management decisions were made on individual workstations. Each workstation implemented its own voice services and managed recorded voice locally. In addition to the speaker built into the computer, the local telephone set could be used as a speaker and microphone by the computer.

Phonetool

Phonetool is a telephone dialing tool that allows users to place calls from their workstation; it was developed by Stephen Casner at the University of Southern California's Information Sciences Institute. Phonetool runs under the Sunview window system and uses a centralized telephone server to place calls. Phonetool provides a pop-up speed-dial menu (see Figure 11.7), can redial the most recently dialed number (which is displayed on the telephone icon), accepts numerical input from the keyboard, and can dial a number displayed in another window using mouse-based cut-and-paste interaction.

Phonetool places calls using a centralized server; a server is required because of hardware interface limitations for access to the telephone switch (a Rolm PBX). The server can place outgoing calls but it is not provisioned to receive them. Software was written to allow a personal computer equipped with a special PBX interface card and an Ethernet controller to function as a centralized server, i.e.,

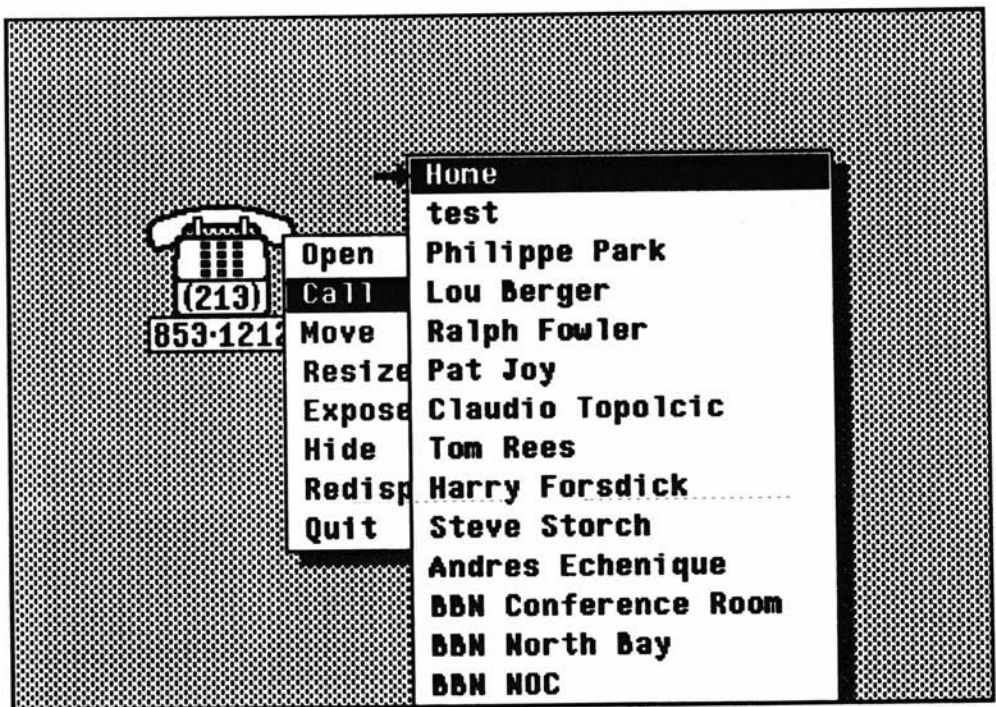


Figure 11.7. Phonetool's speed dialing menu.

by dialing on behalf of the numerous workstations (see Figure 11.8). In order to dial, the user's workstation sends a message over the Ethernet to the server specifying the number to dial. The server then dials the call on its own outgoing phone line. At this point, the server puts the outgoing call on hold and dials the user's intercom extension. Because the intercom is used, this call can go through automatically, putting the called party's telephone into speakerphone mode. At this

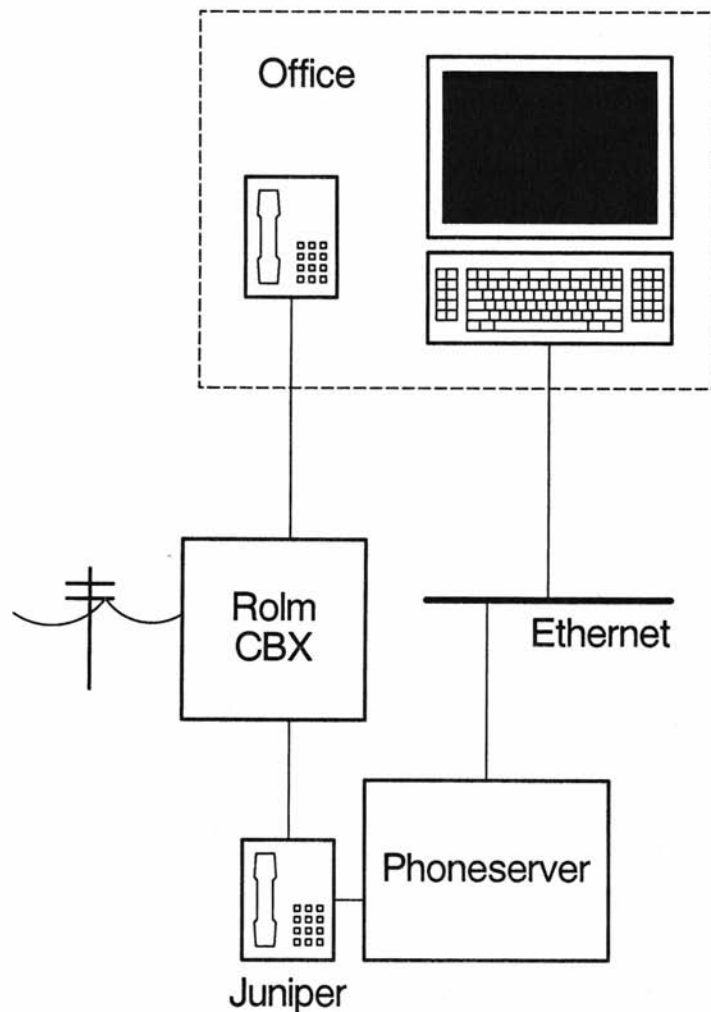


Figure 11.8. The I.S.I. phoneserver utilizes a personal computer with a specialized interface card to control the PBX. Workstations make requests of this server over a local area network. (From Schmandt and Casner, "Phonetool: Integrating Telephones and Workstations." Reprinted with permission from *IEEE Global Telecommunications Conference* © 1989, IEEE.)

point, the server puts the intercom call and the outgoing call together as a conference call and proceeds to drop out of the conversation. The server is now ready to service another request.

Phonetool was the starting point for Xphone described later in this chapter. Separating call control functions from the user interface allowed Xphone to be configured for both centralized and distributed architectures in a number of implementations.

ARCHITECTURES

Several different system architectures can provide telephone functionality to workstations. It is useful to differentiate the *hardware* interface, which physically connects the computer to the telephone switch, from the *functional* interface, whereby applications on the desktop implement telephony services. A **distributed** hardware architecture requires each computer to have a hardware interface for communication with the switch. A **centralized** architecture uses a single link to the switch from one workstation; other workstations access this hardware over the network by communicating with a server process on the specially equipped workstation. Even when a hardware interface is found on each workstation, a local server may be used to allow multiple client applications to share hardware access. All these architectures could offer identical functional interfaces to clients for call control; what differs is the mechanism whereby client requests are communicated to the switch.

Distributed Architectures

Enhanced telephone sets on every desk constitute the most direct implementation of a fully distributed telephone hardware architecture (see Figure 11.9). Advanced telephones may include improved displays, keyboards, touch screens, or speech recognition. These telephones provide superior user interfaces to traditional telephone functionality provided by the switch, and may implement new services, such as electronic mail storage within the telephone. This is not a new concept; in the early 1980s a number of “teleterminals” appeared providing small CRT displays and keyboards. More recent teleterminals use LCD or plasma displays with features such as programmable function keys, stylus input for hand writing, and local voice and text storage for messages. The most recent generation of enhanced telephones is ISDN-based; the ease of data communication between telephones allows for a range of new message services such as digital facsimile and screen-based remote data access.

This approach sidesteps the problem of computer integration, making the telephone powerful enough to obviate the need for the computer. But such devices have not proven popular beyond such consumer products as the combined telephone and answering machine. When made powerful enough to provide support for sophisticated teleservices, such stand-alone telephones become too expensive. Business users already use personal computers or workstations at their desks,

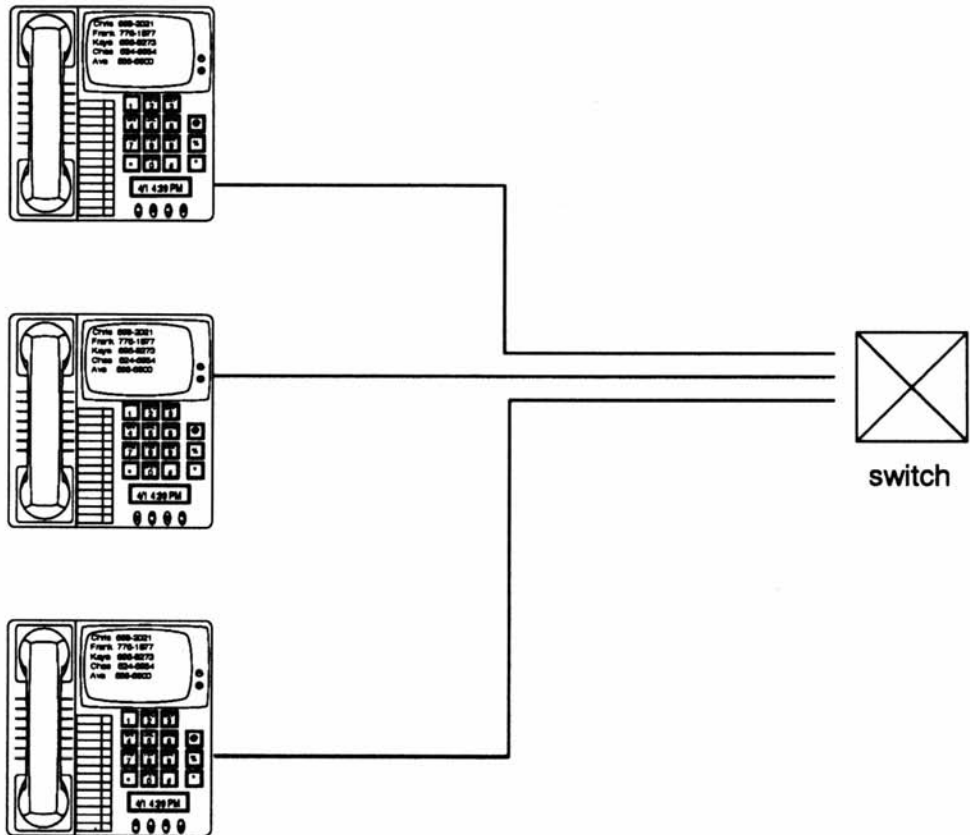


Figure 11.9. A distributed architecture of enhanced telephones.

and the general-purpose computer provides for more efficient resource allocation. Although much can be done to improve the design of telephones, they are not destined to push computers off the desktop. However, in environments where computers are not found, the introduction of ISDN may create a niche market for enhanced telephones.

The teleterminal provides a completely distributed architecture with one telephone per office. A completely distributed architecture can also be used for workstation-based telephony. For example, an inexpensive modem wired in parallel with a telephone can be used with autodialing software; the computer places the call and the user picks up the handset if the call is answered. Similarly, new consumer products plug into an analog phone line and decode calling-line ID information; when this information is available, the calling number is transmitted over a serial line to a host computer. Such approaches are practical on analog lines because modems are inexpensive and computer serial ports ubiquitous, but they provide only limited signaling functionality. For the computer to talk or listen to the voice circuit during a call, a higher degree of integration is necessary.

One means of providing both call control and telephone audio to a personal computer is a specialized telephone interface board. Such a board includes the required electrical interface to allow it to be connected to the telephone network, touchtone generator and decoder integrated circuits, a codec, and perhaps a digital signal processor or an integrated circuit for speech compression. Telephone interface boards are currently available for as little as a few hundred dollars.

Many workstations and personal computers now support audio digitization and playback. Telephone functionality can be added using simple external electronics. An external unit provides a telephone jack, a serial interface for call control, and analog audio input and output to be connected to the workstation. The computer sends commands and receives notification of events such as an incoming call over the serial interface. When the phone line is off hook, the computer's audio input and output are connected to the line. Touch tone dialing can be accomplished by playing prerecorded touch tone sounds once the telephone has been taken off hook. Tone decoding can be performed in the workstation by software (if the workstation is powerful enough) or by an integrated circuit in the external hardware.

ISDN is very attractive for workstation integration of telephony because it provides finer-grained call control due to its richer signaling protocol. The appropriate electrical hardware interfaces are beginning to appear as standard equipment on workstations, with ISDN call control protocols implemented in software. Once a telephone connection is established, the voice channel is already digital so further hardware is not required for voice interaction. Current advanced workstations can easily run Layer 3 ISDN protocols in software under multitasking operating systems. But less powerful computers or operating systems require separate microprocessor-based peripheral hardware to handle the real-time requirements of ISDN. Such peripherals are currently selling in the \$800 range, which may represent a significant portion of the cost of the computer itself.

Centralized Architectures

In the distributed architecture just described, telephone interface hardware is located on every computer. Such an arrangement may be impractical or inadequate for some applications, however. For either proprietary or ISDN digital switches, telephone interface hardware may be prohibitively expensive. Computer reliability is also an issue; when the workstation is down, does the telephone still work? Although it is easy to configure an analog line control card with a second telephone jack to let an adjunct telephone function when power is turned off, this is much more difficult under ISDN signaling. Some PBXs offer sophisticated computer interfaces but are equipped to connect only a single computer (usually through a serial interface); this precludes direct switch access from each workstation.

For any of these reasons, a centralized hardware architecture may be preferred or required. Figure 11.11 illustrates an example of this approach; using a telephone server provides distributed access to a network of computers. One computer, the server, includes the required hardware and software to communicate with the telephone switch on behalf of its clients. Clients are distributed over a