SIX

WRITING EFFECTIVE PROMPTS

A writer is somebody for whom writing is more difficult than it is for other people.

—THOMAS MANN

Writing is easy. All you do is stare at a blank sheet of paper until drops of blood form on your forehead.

—GENE FOWLER

The difference between an adequate speech-recognition system and a great speech-recognition system lies in how the system asks questions and conveys complex information. Great systems do it with an elegance worthy of a haiku; their meaning and impact are clear and immediate, and not a single word is wasted. The more elegant a system is, the more intuitively—and quickly—a caller can use it, and the greater value it offers to both clients and callers.

This chapter does not discuss topics such as "the *best* way to ask a 'yes/no' question." Why not? Because, for one thing, there *is* no single "best way." Rather, there are many different ways to properly ask a "yes/no" question, depending on the situation. I've avoided absolute rules—and words like "always," "never," "best," and "worst"—because the state of the art of design is constantly changing. Any absolute rules I could offer would soon be outdated. In fact, I prefer that people understand and extract the underlying ideas about the design of successful speech systems rather than blindly follow a set of rules.

THE LANGUAGE OF ASKING QUESTIONS

At its most basic level, the design of speech-recognition system prompts is composed of two elements:

- Questions the system asks
- Statements the system makes

The questions that are asked need to be clear and not easily misinterpreted, and the statements need to provide useful information, in a valuable format. And most of the time there is a mixture of statements to inform the caller, and questions by which the system will glean new information, in the same context. This can get tricky when the conversations between the computer and the caller start to get complex. Most people don't realize how much error correction they do in *real* conversations. How often do our conversations with real people sound anything like the carefully scripted conversations of a TV drama? On TV and in movies, characters never say "Uh, what? I didn't get that last thing you just said."

In real life, real people do lots of error checking and correcting. And when we're asked to clarify something we've said, what do we do? We usually rephrase it or provide further information. If the people to whom we're talking need further clarification, we might even explain why we're asking the question.

Speech-recognition systems need to emulate this behavior, because—just as in real conversations—mistakes and misunderstandings do occur. Think of all the things that could possibly go wrong when a system asks a caller a question.

- The caller may not respond.
- The caller may respond with the wrong answer and the recognizer can't understand the response.
- The caller may respond with the right answer, but the recognizer isn't sure enough to accept the response without confirming it first.

To handle all of these possible situations, speech-recognition systems often use these six types of prompts:

- Initial
- Retry
- Timeout
- Help
- Success
- Failure

Let's take a close look at each of these prompt types by going through the types of prompts that make up a typical call flow.

Initial Prompt

When a system enters in a particular state it may provide some information, but then, almost always, it asks a question such as "When do you plan to arrive?" The prompt played to the caller is called (logically) the **initial prompt** since it's the first thing the system says in a particular state/context. The recognizer then listens for the caller's answer.

Retry Prompt

If the recognizer hears something, but can't match the caller's utterance with the command vocabulary it is using in that context, then it will play a **retry prompt**. For example, if a caller says, "Uh, I guess I'm not sure" when the system is expecting to hear a date, such as "April 24th," the system would most likely not understand the utterance and then play a retry prompt to aid the caller.

We often use a retry prompt to clarify an idea with the caller, in the hope that by giving the caller another attempt he or she will understand how to answer a question correctly. For example, after hearing "Uh, I'd like to arrive on the last Saturday in April," an

appropriate retry prompt might be an explicit statement, "I'm sorry I didn't understand. Please say the date you'd like to arrive. For example, you could say 'January 29th' or 'February 2nd.' Or for more information, say 'Help.'"

This retry prompt would help many callers, but not necessarily those who are saying the right thing but aren't being heard well—possibly (increasingly) because of bad connections on mobile phones. If the system still doesn't understand what the caller is saying, it might employ a second retry prompt, similar to the first one but with more or different information. For example, it might tell the caller, "I'm sorry, I still didn't understand. Say the date, or enter it, using touchtones. For example, you'd enter January 13th as zero, one, one, three. Otherwise, say 'Help' for more information."

Currently most speech-recognition engines are designed to reject a response after they've calculated that the caller's response is likely invalid. For example, if the caller answered the question "On which date?" with "The last Saturday in April," most likely the recognizer wouldn't understand that response. However, if the system were somewhat certain—but not certain enough—that it understood the answer, it might ask a confirmation question to verify the response, such as "I think you said April 2nd, is that correct?" This technique saves callers from repeating an answer in a case where the response the recognition engine heard is just below the threshold of acceptance.

Timeout Prompt

The system needs to determine whether the caller responded. If the recognizer doesn't hear anything, it will usually ask another question providing more information to the caller to elicit a response. This is called a **timeout prompt**.

To design timeout prompts we always need to ask ourselves, "Why wouldn't a person respond to this question?"

- Were they distracted and didn't hear or understand the whole question?
- Did they not know the appropriate words to use in response to the question?
- Did they not know the answer to the question?

- Did they not understand why they had to answer the question in the first place?
- Or did they not want to answer the question? Perhaps because they didn't want to reveal sensitive information?

So, for example, if a system asked, "When do you plan to arrive?" and fails to hear a response at all, it can provide a timeout prompt, such as "I'm sorry, I didn't hear you. Say the date that you plan to arrive, or for more information say 'Help.'" This new prompt provides a little more information about how to answer the question by instructing the person to say a date (rather than a relative time marker such as "After I get to Portland") and steers them to say "Help" if they need to find out more information about this question.

If callers' responses aren't heard twice in a state, a second timeout prompt can be used—often to instruct them on how to use touchtones or how to talk to a representative.

Help Prompt

Not to be confused with general help statements that explain how to use the system, the type of **help prompt** to which we're referring gives the caller assistance with a particular task in a particular situation. Designing a help prompt can be something of a challenge, because we, as designers, don't always know why callers would need help in a particular situation. So we have to make educated guesses by asking ourselves, "Why would callers not be sure how to proceed from here?"

Here are a few answers to that question.

- They didn't mean to be there in the first place.
- They changed their minds.
- There was something important missing from the original question.
- They didn't wait to hear the retry or timeout prompt messages.

We need to write our help prompt to address all of these possible needs.

Success Prompt

A **success prompt** is played when a caller has successfully exited the current state and is entering the next state. You may not choose to always use a success prompt, but if you do it can be as simple as "Got it," or more of an informative segue, such as "OK, now let's collect the last piece of information."

Failure Prompt

A **failure prompt** is played when callers have tried—and failed—to answer a question several times. They're obviously stuck and need to be either transferred to a "live" representative in a call center, or re-oriented (perhaps by being returned to the main menu). A typical failure prompt goes something like this: "Sorry, but it seems we're having problems. Let me transfer you to someone who'll be able to help. Hold on." In the case of a client without a call center behind the application, it might say "It looks like were having problems. Let's just take things from the top," then proceed to the main menu.

Often the designer can define this prompt once in the Design Specification and simply call for its use whenever a caller fails at a state. However, the designer should always look for situations in which this prompt may be inappropriate—or inadequate. For example, if the system has already collected information from the caller and can pass it along to a customer service representative, an appropriate prompt might be "Sorry, but it seems we're having problems. Let me transfer you to someone who'll be able to help and I'll pass along the information I've collected so far. Hold on."

THE ART OF WRITING PERFECT PROMPTS

The art of writing the perfect prompt is to convey ideas clearly and concisely. No extra words. No fuzzy language. A well-constructed prompt guides the caller to say *only* things the recognizer will understand. If a recognizer is able to understand almost everything that a caller might say in a particular context, then the prompt can be less directed. For example, a system might just say "Hello, and welcome to Thrifty Car Rental. What would

you like?" Callers might say hundreds of thousands of things to answer this question, and thus the recognizer would have to be able to recognize most of them to be successful. Most recognizers don't work this way at the moment. If, however, the recognizer is looking for a particular word (such as a manufacturer of automobiles), the prompt must direct callers to answer specifically. For example, by having the prompt ask, "What's the automobile manufacturer?" instead of "What type of automobile?" we could minimize the chance that callers would say words such as "van" or "sedan."

In addition, prompt language must be consistent with:

- The original concept of the design
- The way the company expresses itself in other media
- The rest of the text spoken in the application

Writing Effective Initial Prompts and Commands/Command Phrases

Initial prompts must convey ideas clearly so callers can understand what they are expected to say. This doesn't mean that we need to provide callers with a long, drawn-out explanation at every turn of the dialogue. Once we establish that a particular calling population is comfortable with a speech-recognition system—and once we confirm that the caller has successfully answered some basic questions, like "Is that correct? Yes or no?"—then we can start to drop the "Yes or no" part and just ask the basic question.

As we've already discussed, it's important to be clear and consistent when writing prompts that present a list of options. For example, it may not be a good idea to have an initial prompt that says:

"Main menu. You can say 'Reports,' 'Exchange,' or 'Security.'"

The problem? There might be little or no context for the options—and the words themselves offer few clues. Both "Reports" and "Exchange" can be either nouns or verbs, and "Security" could refer to anything from data encryption to insurance coverage.

If, however, we rewrote the prompt so that all the selections were expressed in the form of imperative sentences that provide greater precision, callers could quickly understand the ideas—and their responses would also feel more conversational:

"Main menu. You can say 'Get the reports,' 'Make an exchange,' or 'Change my PIN.'"

Consistency is always important when designing prompts or when those prompts contain commands that the caller is to use later on. For example, the Wildfire Communications "voice-activated personal assistant," also called "Wildfire," uses the command "Describe it" to enable users to hear information about their phone messages (rather than listen to the actual message), such as the time the call arrived. But the clever thing about the design is that the user can say "Describe it" for any object—and Wildfire will tell the "header" information about that object. If a user said, "Describe it" while working with an e-mail message, Wildfire would tell the caller additional information about that message, such as who sent it, the subject of the message, and so on. By teaching the user a single command—and employing it consistently throughout the application—Wildfire makes it easy for its users to get more out of the system without learning new commands.

Designing Effective Retry and Timeout Prompts

How many retry and timeout prompts should a system give a caller before considering it a failure? I generally recommend presenting no more than two; anything more is likely to irritate a caller.

While it's sometimes acceptable to repeat the same prompt twice, often the designer will want to vary them to give the caller more information to help them answer the question correctly or get them back on track. As with help prompts, the best way to begin writing a retry or timeout prompt is by answering the question "Why would the caller not have said something valid at this point?" By considering all the possible answers to that question, we can write prompts that address most—if not all—of them. Many of the possible answers relate back to examining how the original prompt was written. For example,

if we were designing a system for a motel by an interstate highway (frequented largely by travelers arriving by car or truck), we might assume that we needn't ask "Will you be parking a vehicle in our lot?" Instead, we'd skip ahead and ask, "What's the make and model of your vehicle?"

However, there will be at least a few people who won't be traveling to the motel by car or truck, but by bus, taxi, or maybe even on foot. The question "What's the make and model of your vehicle?" does not apply to them—and they wouldn't know how to skip the question. A retry or timeout prompt could circumvent this by saying "If you aren't bringing a vehicle, just say 'Let's go on.' Otherwise, what's the make and model of your vehicle?"

A similar situation arises when a caller embarks down a path they either didn't mean to take, or no longer wishes to take. A retry prompt can get the caller *unstuck* by including the phrase "...or if you don't want to be here say 'Main menu.'"

To be more foolproof, retry and timeout prompts should include further information about how to answer the question, but not so much information that it overwhelms the caller. Often the easiest way to do this is by providing a quick example. Instead of "Say the model year and make of your vehicle," the retry prompt could be "Say the model year and make of your car; for example, '1969 Mercedes-Benz.'" This tells the caller what format to use in answering the question.

An alternative is to suggest the touchtone equivalents to spoken answers, as in this example: "You can say 'Get the reports' or press 1, 'Make an exchange' or press 2, or say 'Change my PIN' or press 3." This approach is obviously better suited to questions with a limited number of possible responses—not hundreds, as in the vehicle example above.

Finally, all retry and timeout prompts should offer to direct callers to either the help prompt or a "live" representative, as in this example: "You can also say 'Help' or press the star key, or press zero for an operator."

Designing Effective Help Prompts

The help prompt is often the hardest to write, because the sequence of events leading up to the situation isn't tracked. Did the caller say "Help" immediately after hearing the initial question—or after two timeout and two retry prompts? We could be dealing with callers in greatly varying states of mind—from mildly confused to frustrated and irate.

When writing help prompts we again ask ourselves, "Why would anyone *not* know how to answer this prompt?" We need to ensure that the caller knows

- Why we're asking the question
- Where they are in the dialogue
- How (e.g., the format) to answer the question correctly
- How to get more help (e.g., talking to a "live" representative)
- How to escape this state to a "safety" zone (e.g., a main menu)

Here's a typical initial prompt and the help prompt that went with it.

***Initial Prompt*: At what airport, or city, are you picking up the car?**

***Help Prompt*: Sometimes, rates and availability can depend on the location where you're picking up the car. Thrifty Car Rental has locations at airports throughout the U.S., Canada, and also in the rest of the world. *<pause>* To specify an airport, just say the name of the city and state you're interested in.**

Note the use of the pause in that prompt. In this context that pause serves as a way to separate the background explanation from the command that the caller should react to—the pause gives a rhythmic break to regain the caller's attention.

Designing Effective Confirmation Prompts

Confirmations are an essential part of any design—even if they seem casual and unobtrusive—and they come in two varieties: explicit and implicit.

An **explicit confirmation** is when the system prompts the caller to answer a direct question before proceeding. For example: "OK, I think you'd like to buy 100 shares of Apple Computer, symbol AAPL, at the market price. Is that correct?"

This type of confirmation is best used when the risk of an unrecoverable error is present. To further prevent mistakes caused by misinterpretation of a caller's response, you can employ one of the following methods.

- The system can ask a "yes/no" question; for example, "Is that correct? Yes or no?"
- The system can ask the caller to say a PIN or a special password to confirm the transaction—or say "Cancel it" to cancel the transaction. This method assumes that the recognizer will not confuse the words "Cancel it" and the caller's password.
- The system can supplement the password confirmation by asking the caller to enter a number using the touchtone keypad to confirm or cancel. For example: "You have requested to purchase 100 shares of Apple Computer, symbol AAPL, at the market price. To confirm this transaction, press 1. To cancel, press 9." (If you use this method, attempt to avoid numbers that are close to each other on the keypad.)
- The system can disable the caller's ability to truncate the playing of a prompt by saying something or entering a touchtone number. If the system is programmed to disable this feature only during confirmations, it will prevent callers from accidentally assuming that the transaction is correct before hearing the entire confirmation statement. This method also reduces the liability of the company in the event of an error because it proves that the company did its best to ensure that the caller heard the entire statement before confirming it. It's worth noting that many companies save recordings of caller transactions to defend themselves in the event of any legal action—usually to defend themselves from callers who claim that the *machine made a mistake* when, in fact, the system actually did exactly what the caller requested.

WRITING PROMPTS FOR ELEGANCE, SPEED, AND VALUE

The difference between an adequate speech-recognition system and a great speech-recognition system lies in how the system asks questions and conveys complex information. Great systems do it with an elegance worthy of Audrey Hepburn; their meaning and impact are clear and immediate, and not a single word is wasted. The more elegant a

system is, the more intuitively—and quickly—a caller can use it, and the greater value it offers to both clients and callers.

There are several mistakes commonly made that negatively affect the elegance, speed, and value of the system: providing unnecessary information, using ambiguous language, and not getting callers to focus on the essentials.

Unnecessary Information

Too often I hear systems that provide all callers with information that might only be useful to a small percentage of them. For example, I heard a cautionary message for U.S.-based services (used predominantly by U.S. residents who only make calls within the country) that provided information useful to only a small population. With some editing to protect the anonymity of the company, the prompt said something very similar to: "There could be a charge associated with using an access number if you call from the U.S. to another country such as, but not limited to, the former Soviet Union, Croatia, and Albania. Now, please tell me name of the city and state in which you live." Of course, sometimes disclaimers such as this are required by law, but they should only be played when necessary—not to every person who calls in. Not only is it long and distracting, but it also adds extra time to the call—and that means higher costs for the company paying for the toll-free line. Also, the disclaimer won't be effective if people are not going to listen to it, so it's in the best interest of the company to allow the designer to work with its legal team to aid in the writing of any required prompts so that the prompts are concise and intelligible while providing value to the listener and covering the company's liability.

Ambiguity in Language

In today's fast-paced world, we all use ambiguous or imprecise language as a kind of shorthand to save time and effort. Imprecise language causes the least negative impact in situations where a context can be clearly established and there is a high bandwidth of information exchange. An example is in face-to-face situations with people we know, where prior knowledge, body language, and vocal cues can fill in the blanks and add meaning to the words.

But it's a different story when one or more of those additional communication components—prior knowledge, body language, and vocal cues—are absent. The most extreme example is e-mail from an unknown person, where all three are absent. As every e-mailer knows, the intended meaning of written messages can easily be lost or misconstrued—particularly sarcasm and other forms of humor—which can lead to misunderstandings. Many e-mailers try to avoid this problem by adding **emoticons** and acronyms to their messages to help clarify their meaning:

"I know we hung out this morning but it feels like forever. :-) "

—where the set of *colon, dash, and close parenthesis* look like a smiley face turned 90 degrees counterclockwise.

Speech-recognition systems have the tremendous advantage of audio—which can include vocal cues, sound effects, and music—to add meaning to their outgoing communications. But because speech systems require clear answers from callers in order to function properly, it's essential that all prompts be as concise and unambiguous as possible.

That means we must carefully consider the language we use to convey ideas—even to the point of avoiding language that not only could be *misconstrued*, but even *misheard*. We can examine this situation by examining how people talk to each other. I was at a family gathering recently and overheard the following exchange among my Uncle Rob, Aunt Bobbie, and 22-year-old cousin Paul.

UNCLE ROB: ***Hey, Bobbie—I just found out Paul isn't dating a woman six years older than him—she's only six days older than him.***

AUNT BOBBIE: ***Well, that's certainly a relief.***

COUSIN PAUL: ***Uh, guys? You're both wrong. She's actually six months older than I am.***

The misunderstanding arose because my uncle misheard the age difference, only remembering the "six." Had my cousin said, "Yeah, she's half a year older than I am," there would have been less chance for confusion (and alarm on my aunt's part!). Using the phrase "half a year" instead of "six months" would limit the number of ways a listener could

reasonably mishear the sentence. And none of those possible misheard phrases ("half a day," "half a week," or "half a year") would likely cause alarm—even to my Aunt Bobbie.

We can do the same thing when we write prompts for a speech-recognition system by understanding—and avoiding—the ambiguities of real-world conversations, and steering clear of language that could be misheard or misconstrued. Some systems try to avoid errors by repeating information back to the caller. This is a good idea when an answer needs to be precise—such as a stock trade—but unnecessary when a precise answer is not essential or can be easily modified later on.

GETTING CALLERS TO FOCUS ON THE ESSENTIALS

It's important for a system to use precise language, but other design components can make it easier for callers to comprehend and use the system—in particular, the order in which ideas are presented, and how those ideas are presented.

Presenting Information Clearly and Usefully

If we were designing a system that provides a warning message, we would want the system to alert the caller first before playing the message. Here's an example of a message that gets played when a single stock-trading account is accessed by more than one person at the same time. The system needs to alert the callers that this activity could be the result of an intruder.

> **"Sorry, but the system is alerting me that there is another person accessing this account right now. I've alerted the system administrator, and for security protection, I'm disabling some functionality for this account (in case it isn't you on another phone), such as making a trade, reporting balances, and other things. However, you can still get real-time quotes and news updates."**

By designing the system this way, we enable callers to decide whether they want to focus on the warning message and act upon it, or simply let it play so they can move on.

Presenting Information in a Meaningful Order

Some people apparently don't know the difference between essential and nonessential information. I'm sure we've all been at social gatherings where someone has trapped us in a corner with the promise of a fascinating anecdote, only to see it turn into a minute-by-minute, detailed account of his or her day. This forces us to try to filter out the unimportant data on the fly—which can be exhausting (and often fruitless). We feel like yelling, "Get to the point—or let me out of here!"

The same holds true for speech-recognition systems. Even if we think it's obvious what's important, we can't assume that callers will be able to filter those nuggets out of the rock pile. We can see how we modify our behavior in real life—as when I talk to my grandmother.

If I said to her:

> *"I walked past the yellow house on Main Street, and the first store on the right, which is a shoe store, has a pair of shoes in the window costing $45 that I think you'd really like."*

... she might give every detail of that account equal weight. She'd probably ask questions about the yellow house or its position on Main Street in order to clarify things in her mind, instead of focusing on the part I wanted her to—the inexpensive shoes. I should say:

> *"Grandma, I saw an inexpensive pair of shoes I think you would really like! If you're interested, just walk to the shoe store on the right side of Main Street (close to the yellow house) and look in the window for shoes marked $45."*

This construction uses the first sentence to set the context, and the second sentence to focus the details on how to achieve the goal. If Grandma didn't want a new pair of shoes, she could completely disregard the second sentence, knowing that it was only there to support the first.

Generally, the most important information should be presented first, and by *important* we mean information that is either the most critical or most descriptive of the context.

Here's an example of how a designer could make a mistake in a speech-recognition system.

"Flight 534 departing from Chicago O'Hare today, Thursday, July 5th from gate B9 in terminal 1, concourse B, is currently scheduled to depart at 8:45 P.M., on time."

In this example the most important information—the status and time—is buried deeply in the prompt. A better approach would be to guide the caller's focus to the most important information at or close to the beginning of the statement, hierarchically, so that if the information isn't immediately relevant to them they can ignore the rest of the statement.

"Flight 534 is scheduled to depart on time, at 8:45 P.M. from Chicago O'Hare today, Thursday, July 5th from terminal 1, concourse B, gate B9."

All callers need to know the status of the flight, and secondly, the departure time. Callers who are frequent flyers probably know the terminal number of the airline, and potentially even the concourse, then just read the gate information from the monitors. The information they need to know when calling (perhaps on their way to the airport) is whether the flight is on time. It's not good to bury important information deep inside the prompt, particularly if the plane is going to be delayed for four hours, in which event the gate information has a higher probability of changing.

These examples are a good way to understand some of the tricky elements of designing effective systems. However, it's necessary to get all the elements together to form the Design Specification from which the actual system will be produced.

SOME SUBTLETIES OF PROMPT WRITING

Some words and sentence constructs work better than others in speech-recognition systems—even when both are grammatically correct. Contrary to the protests of millions of

elementary schoolchildren over the years, there are many reasons why we should speak using proper grammar—even in speech-recognition applications. The two biggest reasons? Precision and clarity. Grammatically correct language (unless it sounds extremely awkward) leaves less opportunity for misunderstanding.

I'm not overly pedantic about correct grammar and usage—it can be taken to ridiculous extremes—but a disregard for language betrays a certain sloppiness or lack of attention that reflects on a person or a company. For example, why do most supermarkets have checkout lines incorrectly labeled "12 items or less," instead of the correct "12 or fewer items?" "Fewer" has only one more letter than "less," so they're apparently not doing it to save space on their signs.

We can imagine a vacation-marketing survey system that asked questions about how people travel, and how they have enjoyed particular vacations booked through this company. If the system asked a series of questions about the person who accompanied the caller on a recent trip, the system might ask, "With whom were you?" This question, though grammatically correct, would confuse many people, and perhaps should be worded more colloquially as "Who were you with?"—to ensure that most people would understand the question.

Here are some rules of thumb that apply to virtually all U.S. applications.

"Want" Not "Wish"

Unless we're talking about systems run by genies or fairy godmothers, it's preferable to use "want" instead of "wish." People want answers, they don't usually wish for them. So instead of "Do you wish to search by date or by price?" use "Do you want to search by date or by price?"

"Say" Not "Speak"

"Enter or say your password" sounds a lot more natural than "Enter or speak your password." "Speak" sounds like a clinical term ("Yes, Doctor, the subject speaks whenever the bell rings. He also salivates.") The word "say" conveys a softer and more natural idea.

Contractions

Designers should use contractions in their prompts. Of course, when some people write text, they often do not use contractions, and it is perfectly correct. But try reading the sentence preceding this one out loud. It sounds as stilted and unnatural as the ending of this sentence, does it not? One of the great advantages of a speech-recognition system is its ability to create an affinity between the company and the caller—and it's harder to establish that sense of familiarity and comfort if the prompts are devoid of contractions, because most people simply don't talk that way. People are more likely to use—and enjoy using—a system if they feel there's a *regular* person on the other end of the line (even if that person happens to be a machine), and contractions help create that sense.

Word Order Matters

Often a sentence can be constructed in several ways, all of them grammatically correct. Which construct should we use? Whichever one more precisely conveys the idea. For example, the following two statements are both grammatically correct, but while the first correctly conveys the idea, the second could cause callers to start forming an incorrect mental model.

"If you don't think I'm going to get it right, say "Help.'"

"Say 'Help" if you don't think I'm going to get it right."

The first sentence correctly indicates that in a situation where callers don't think the system is getting it right, they can say "Help" to (we would imagine) get a better understanding about the situation. In the second sentence, callers are instructed to say "Help" if—and perhaps only if—they think the computer won't get it right. If the word "Help" can be used in multiple contexts, we don't want to limit its use to only one of them.

The other reason why the first sentence is preferable to the second is that the caller's action—to say "Help"—is revealed *after* the system describes the circumstances that would prompt the action. This is a more logical sequence, and since callers' memories are short, it's always better to put the most important part of the instruction—to say "Help"—at the end.

Use of the Word "Just"

According to my dictionary, the word "just" has 13 meanings in English—6 as an adjective and 7 as an adverb. The differences in these meanings can be significant. For example, consider these two uses of "just."

	Meaning of "just"
Add just enough salt to give it flavor.	**precisely, exactly**
To get assistance, just say "Help."	**simply, merely**

In a speech system, we could have the system say either

"You say the search topic, and I'll look for something that sounds like it."

or:

"Just say the search topic, and I'll look for something that sounds like it."

The first sentence indicates that "anything you say" will be considered the search topic. The second statement is intended to say that the user simply needs to say a word to get the system going. However, a caller could misconstrue the meaning of "just" to be as it is in the first sentence above. Under that meaning, it sounds as if callers are required to know a precise search topic word (apparently from some top-secret list that the system isn't sharing) to get any results.

Use of "Want," "Like," "Can," and "May"

These words are often used interchangeably, but they actually have different meanings. Consider the following questions.

"Do you want me to read that back to you?"

"Would you like me to read that back to you?"

"Can I read that back to you?"

"May I read that back to you?"

The first question can evoke several reactions, depending on how it's said. It could sound as if it's urging the caller to listen, as in ""You *do* want me to read this to you, don't you?" However, it can also be directed to sound like the most neutral way to suggest the idea, as in "Do you want me to do this? Because I really don't care if I do or not."

By using "would," the second question sounds a little more formal and deferential than the previous statement, and it reinforces the personality of the system as someone eager to be helpful.

The third question is grammatically incorrect, substituting "can" for the correct "may." If taken literally, it means "Am I able to read that back to you?"—an irrelevant question. Beyond the grammar problem, it sounds as if the system really *wants* to read the statement back and is only waiting for the caller to give in and let it.

While "May I read that back to you?" is grammatically correct, it sounds as if the system is pleading with the caller to let it read the statement again. For some reason, "may" sounds too contrived to me. I have visions of a very proper British valet saying "May I draw your bubble bath now, your Lordship?" That's why I avoid it when designing any application.

Natural Language Shortcuts

A natural language shortcut allows callers to "skip" a bunch of steps by allowing them to provide several pieces of information to the application all in one sentence. The use of natural language shortcuts can be very effective, but only if the caller knows how—and where—to use them. For example, take a look at this exchange.

SYSTEM: **Where do you want to fly from?**
CALLER: ***Boston.***
SYSTEM: **Where do you want to fly to?**
CALLER: ***San Francisco.***
SYSTEM: **At what time?**
CALLER: ***3 P.M.***

This exchange could be shortened considerably if the recognizer understood a natural language shortcut that allowed the caller to provide all the information (while still being able to understand just the first piece of information—in this case, the departure city).

SYSTEM: **Where do you want to fly from?**

CALLER: ***From Boston to Los Angeles, at 3 P.M.***

This is a very convenient, quick, and natural option for people, but it is difficult for some recognizers to handle such a complex task. In fact, the recognition can be seriously compromised if the recognizer is expecting to hear only one token (a single piece of information, such as "Boston, Massachusetts") and instead hears a long string of several tokens (such as "Boston, Massachusetts to LAX, at 3 P.M."). People who tune these systems to improve recognition accuracy can do a lot to prevent recognition problems, but issues can still arise when we allow the use of a natural-language shortcut designed to listen for, and process several tokens while at the same time allowing the caller to simply indicate just one token of information. It's like a person ordering a pizza and just saying, "I'll have a pizza." The cook might expect that the person would indicate the size, and the toppings (even if it's just a "plain" pizza.) People can generally cope with this situation, but even real people get a little confused for a moment when they expect to hear more information than the person provides.

But what if a system does allow natural language shortcuts? How does it teach callers to take advantage of it? Is it best to tell all callers that they can "Just say their flight itinerary?" Or should that instruction only go to repeat callers who have a better chance—and probably a greater need and incentive—to memorize the structure that the system is looking for?

If the system has been programmed to identify callers and track their usage, it can be personalized to work in whichever way is best for each individual. For example, in a stock trading application, the system could have the caller work through a series of steps:

SYSTEM: **What kind of trade do you want to make? You can say "Buy," "Sell," "Sell short,"—**

CALLER: ***Buy.***

SYSTEM: **How many shares do you want to buy?**

CALLER: *100.*

SYSTEM: **Of which security?**

CALLER: *Chemex Coffee Corporation.*

SYSTEM: **At what price?**

CALLER: *A limit price.*

SYSTEM: **Of what?**

CALLER: *88.*

SYSTEM: **OK, let me confirm that with you ....**

After the system has led the user through this process several times, it can be programmed to say the following (after the trade is completed).

"Here's a hint: Next time, you can say the whole trade when I ask you the first question. So, for example, you could say, 'Buy 100 shares of Chemex Coffee Corporation at a limit price of 88'—all in one breath."

It's usually better to use callers' most recent trade as the example, because it personalizes and makes the experience more concrete and relevant to the callers—all of which will help them remember it the next time they call.

TOP FIVE GOOD TENETS FOR WRITING PROMPTS

1. **State what the application will do and how it will work before engaging callers in conversation. The application needs to set the expectations of the caller and explain the role that it will serve. For example, a rate quote application that is made for people who have never spoken to a computer before would benefit from a phrase like "In a moment, we'll have a short conversation where you'll tell me what you want to ship, and I'll tell you how much it will cost." The caller knows that they'll need to talk to this computer and that the conversation will be short, and the system can provide a shipping rate quote. Sometimes the phrase can be as simple as**

"Welcome to the United Airlines Flight Information System." This phrase is mainly intended to orient the user, and to prevent them from thinking that they can use the application for other services, such as buying a ticket or redeeming frequent flyer miles.

2. Design to the caller's level of knowledge at each state in the application, understanding that callers will learn terms and procedures as they use an application. Each state of an application should not be designed in isolation. As callers learn how to use a system, the system can take advantage of this to either shorten the length of the prompts, or to provide more information.

3. Use a consistent sentence structure for all commands within a single prompt (for example, "[VERB] the [NOUN]"). Other popular structures include

   [VERB] the [NOUN], as in: "'Make a payment,' 'Check balances,' and 'Calculate loan amount'"

   [NOUN/NOUN PHRASE], as in: "'Payments,' "Loan calculators,' and 'Balances'"

   [VERB], as in: "'Delete,' 'Play,' and 'Save'"

4. Ensure that all prompts should have a conversational tone in language and recordings to convey ideas clearly and simply. In general, a well-written prompt doesn't need to have a secondary explanation that attempts to simplify the concept in common parlance. The text should always be natural enough that further explanation of the concept provides *additional information* rather than simply restating the concept using simplified language.

5. Tell callers only as much as they need to know to make effective decisions—no more and no less.[1] Achieving the balance between too much and too little information can be tricky, but start by erring on the side of too little information so that brevity is achieved. Then examine the prompt to determine if there's something that absolutely cannot be left out.

---

1. To learn more about this see H. P. Grice,"Logic and Conversation," in *Syntax and Semantics*, Vol. 3, Peter Cole and Jerry Morgan, Eds. (San Diego: Academic Press, 1975) pp. 41–58.

TOP FIVE MISTAKES WHEN WRITING PROMPTS

1. Getting caught up in the details of the wording of an application before fully understanding its structure. This mistake can lead to lengthy rewriting of prompts if there becomes a need to change the structure of the application.
2. Writing overly verbose prompts.[2] People always consider the *context* when they evaluate a question. It is important to maintain a balance between clarity and brevity.
3. Repeating the initial prompt for the timeout and retry prompts. Timeout and retry prompts should provide additional information to clarify and guide the caller.
4. Using language not commonly found in conversation. For example, "won't" is almost always preferable to "will not."
5. Equating stilted language with formal language. It is possible to be both formal and simple. Instead of saying "You are required to provide your date of birth before we can proceed," you might choose to say, "We need to know your date of birth so that we can find your records."

WHERE WE'VE BEEN—WHERE WE'RE GOING

When designers finally finish getting their ideas on paper, they need someone to produce their work. That's where the producers come in. Showing someone a Design Specification is one thing, but actually using a real system is something else.

2. To learn more about this see Paul Grice, *Studies in the Way of Words* (Cambridge, MA: Harvard University Press, 1989) pp. 30–31.