An Introduction to Speech Recognition

Kimberlee A. Kemble
Program Manager, Voice Systems Middleware Education
IBM Corporation

Have you ever talked to your computer? (And no, yelling at it when your Internet connection goes down or making polite chit-chat with it as you wait for all 25MB of that very important file to download doesn't count). We mean, have you really, really talked to your computer? Where it actually recognized what you said and then did something as a result? If you have, then you've used a technology known as speech recognition.

VoiceXML takes speech recognition even further. Instead of talking to your computer, you're essentially talking to a web site, and you're doing this over the phone.

OK, you say, well, what exactly is speech recognition? Simply put, it is the process of converting spoken input to text. Speech recognition is thus sometimes referred to as speech-to-text.

Speech recognition allows you to provide input to an application with your voice. Just like clicking with your mouse, typing on your keyboard, or pressing a key on the phone keypad provides input to an application, speech recognition allows you to provide input by talking. In the desktop world, you need a microphone to be able to do this. In the VoiceXML world, all you need is a telephone.

For example, you might say something like "checking account balance", to which your bank's VoiceXML application replies "one million, two hundred twenty-eight thousand, six hundred ninety eight dollars and thirty seven cents." (We can dream, can't we)?

Or, in response to hearing "Please say coffee, tea, or milk," you say "coffee" and the VoiceXML application you're calling tells you what the flavor of the day is and then asks if you'd like to place an order.

Pretty cool, wouldn't you say?

A closer look...

The speech recognition process is performed by a software component known as the **speech recognition engine**. The primary function of the speech recognition engine is to process spoken input and translate it into text that an application understands. The application can then do one of two things:

- The application can interpret the result of the recognition as a command. In this case, the application is a **command and control** application. An example of a command and control application is one in which the caller says “check balance”, and the application returns the current balance of the caller’s account.
- If an application handles the recognized text simply as text, then it is considered a **dictation** application. In a dictation application, if you said “check balance,” the application would not interpret the result, but simply return the text “check balance”.

Note that VoiceXML 1.0 uses a command and control model for speech recognition.

Terms and Concepts

Following are a few of the basic terms and concepts that are fundamental to speech recognition. It is important to have a good understanding of these concepts when developing VoiceXML applications.

Utterances

When the user says something, this is known as an **utterance.** An utterance is any stream of speech between two periods of silence. Utterances are sent to the speech engine to be processed.

Silence, in speech recognition, is almost as important as what is spoken, because silence delineates the start and end of an utterance. Here's how it works. The speech recognition engine is "listening" for speech input. When the engine detects audio input - in other words, a lack of silence -- the beginning of an utterance is signaled. Similarly, when the engine detects a certain amount of silence following the audio, the end of the utterance occurs.

Utterances are sent to the speech engine to be processed. If the user doesn't say anything, the engine returns what is known as a silence timeout - an indication that there was no speech detected within the expected timeframe, and the application takes an appropriate action, such as reprompting the user for input.

An utterance can be a single word, or it can contain multiple words (a phrase or a sentence). For example, "checking", "checking account," or "I'd like to know the balance of my checking account please" are all examples of possible utterances - things that a caller might say to a banking application written in VoiceXML. Whether these words and phrases are valid at a particular point in a dialog is determined by which grammars are active (See "Grammars"). Note that there are small snippets of silence between the words spoken within a phrase. If the users pauses too long between the words of a phrase, the end of an utterance can be detected too soon, and only a partial phrase will be processed by the engine.

Pronunciations

The speech recognition engine uses all sorts of data, statistical models, and algorithms to convert spoken input into text. One piece of information that the speech recognition engine uses to process a word is its pronunciation, which represents what the speech engine thinks a word should sound like.

Words can have multiple pronunciations associated with them. For example, the word "the" has at least two pronunciations in the U.S. English language: "thee" and "thuh." As a VoiceXML application developer, you may want to provide multiple pronunciations for certain words and phrases to allow for variations in the ways your callers may speak them.

Grammars

As a VoiceXML application developer, you must specify the words and phrases that users can say to your application. These words and phrases are defined to the speech recognition engine and are used in the recognition process.

You can specify the valid words and phrases in a number of different ways, but in VoiceXML, you do this by specifying a **grammar**. A grammar uses a particular syntax, or set of rules, to define

the words and phrases that can be recognized by the engine. A grammar can be as simple as a list of words, or it can be flexible enough to allow such variability in what can be said that it approaches natural language capability.

Grammars define the domain, or context, within which the recognition engine works. The engine compares the current utterance against the words and phrases in the active grammars. If the user says something that is not in the grammar, the speech engine will not be able to decipher it correctly.

Let's look at a specific example: "Welcome to VoiceXML Bank. At any time, say main menu to return to this point. Choose one: accounts, loans, transfers, or exit." The grammar to support this interaction might contain the following words and phrases:

- accounts
- account balances
- my account information
- loans
- loan balances
- my loan information
- transfers
- exit
- help

In this grammar, you can see that there are multiple ways to say each command.

You can define a single grammar for your application, or you may have multiple grammars. Chances are, you will have multiple grammars, and you will activate each grammar only when it is needed.

You can imagine that you want to put careful thought into the design of application grammars. They can be as restrictive or as flexible as your users and application needs it to be. Of course, there are tradeoffs between recognition speed (response time) and accuracy versus the size of your grammar(s). You may want to experiment with different grammar designs to validate one that best matches the requirements and expectations of your users.

Speaker Dependence vs. Speaker Independence

Speaker dependence describes the degree to which a speech recognition system requires knowledge of a speaker's individual voice characteristics to successfully process speech. The speech recognition engine can "learn" how you speak words and phrases; it can be trained to your voice.

Speech recognition systems that require a user to train the system to his/her voice are known as **speaker-dependent** systems. If you are familiar with desktop dictation systems, most are speaker dependent. Because they operate on very large vocabularies, dictation systems perform much better when the speaker has spent the time to train the system to his/her voice.

Speech recognition systems that do not require a user to train the system are known as **speaker-independent** systems. Speech recognition in the VoiceXML world must be speaker-independent. Think of how many users (hundreds, maybe thousands) may be calling into your web site. You cannot require that each caller train the system to his or her voice. The speech recognition system in a voice-enabled web application MUST successfully process the speech of many different callers without having to understand the individual voice characteristics of each caller.

Accuracy

The performance of a speech recognition system is measurable. Perhaps the most widely used measurement is accuracy. It is typically a quantitative measurement and can be calculated in several ways. Arguably the most important measurement of accuracy is whether the desired end result occurred. This measurement is useful in validating application design. For example, if the user said "yes," the engine returned "yes," and the "YES" action was executed, it is clear that the desired end result was achieved. But what happens if the engine returns text that does not exactly match the utterance? For example, what if the user said "nope," the engine returned "no," yet the "NO" action was executed? Should that be considered a successful dialog? The answer to that question is yes because the desired end result was acheived.

Another measurement of recognition accuracy is whether the engine recognized the utterance *exactly* as spoken. This measure of recognition accuracy is expressed as a percentage and represents the number of utterances recognized correctly out of the total number of utterances spoken. It is a useful measurement when validating grammar design. Using the previous example, if the engine returned "nope" when the user said "no," this would be considered a recognition error. Based on the accuracy measurement, you may want to analyze your grammar to determine if there is anything you can do to improve accuracy. For instance, you might need to add "nope" as a valid word to your grammar. You may also want to check your grammar to see if it allows words that are acoustically similar (for example, "repeat/delete," "Austin/Boston," and "Addison/Madison"), and determine if there is any way you can make the allowable words more distinctive to the engine.

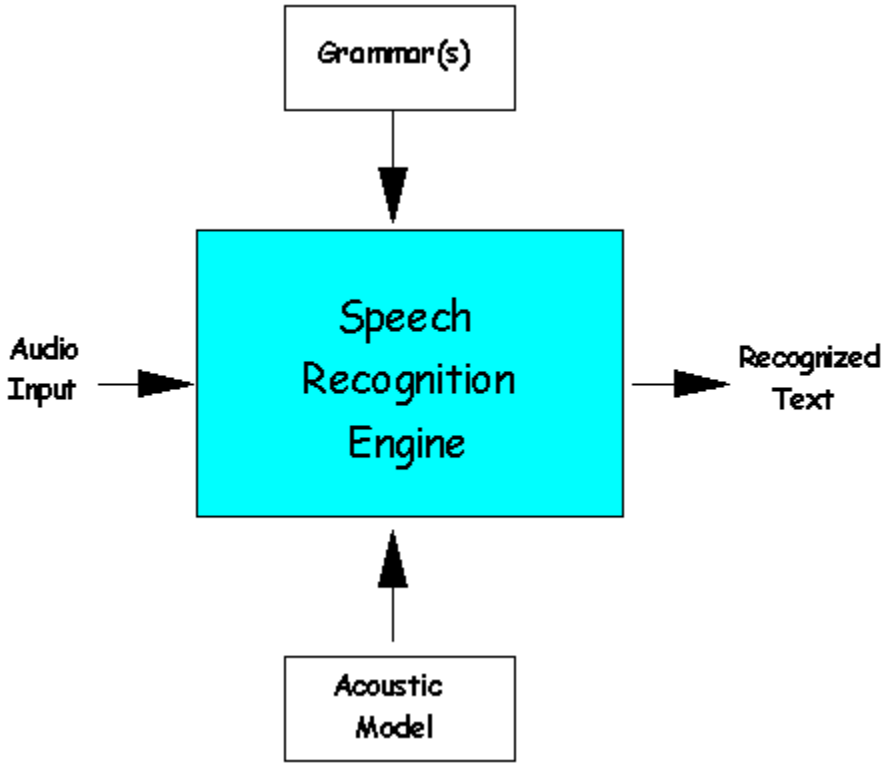
Recognition accuracy is an important measure for all speech recognition applications. It is tied to grammar design and to the acoustic environment of the user. You need to measure the recognition accuracy for your application, and may want to adjust your application and its grammars based on the results obtained when you test your application with typical users.

How it works

Now that we've discussed some of the basic terms and concepts involved in speech recognition, let's put them together and take a look at how the speech recognition process works.

As you can probably imagine, the speech recognition engine has a rather complex task to handle, that of taking raw audio input and translating it to recognized text that an application understands. As shown in the diagram below, the major components we want to discuss are:

- Audio input
- Grammar(s)
- Acoustic Model
- Recognized text



The first thing we want to take a look at is the audio input coming into the recognition engine. It is important to understand that this audio stream is rarely pristine. It contains not only the speech data (what was said) but also background noise. This noise can interfere with the recognition process, and the speech engine must handle (and possibly even adapt to) the environment within which the audio is spoken.

As we've discussed, it is the job of the speech recognition engine to convert spoken input into text. To do this, it employs all sorts of data, statistics, and software algorithms. Its first job is to process the incoming audio signal and convert it into a format best suited for further analysis. Once the speech data is in the proper format, the engine searches for the best match. It does this by taking into consideration the words and phrases it knows about **(the active grammars),** along with its knowledge of the environment in which it is operating (for VoiceXML, this is the telephony environment). The knowledge of the environment is provided in the form of an **acoustic model**. Once it identifies the the most likely match for what was said, it returns what it recognized as a text string.

Most speech engines try very hard to find a match, and are usually very "forgiving." But it is important to note that the engine is always returning it's best guess for what was said.

Acceptance and Rejection

When the recognition engine processes an utterance, it returns a result. The result can be either of two states: **acceptance** or **rejection**. An accepted utterance is one in which the engine returns recognized text.

Whatever the caller says, the speech recognition engine tries very hard to match the utterance to a word or phrase in the active grammar. Sometimes the match may be poor because the caller said something that the application was not expecting, or the caller spoke indistinctly. In these

cases, the speech engine returns the *closest* match, which might be incorrect. Some engines also return a **confidence score** along with the text to indicate the likelihood that the returned text is correct.

Not all utterances that are processed by the speech engine are accepted. Acceptance or rejection is flagged by the engine with each processed utterance.

Speech Recognition in the Telephony Environment

VoiceXML uses speech recognition over the telephone, and this introduces some unique challenges. First and foremost is the bandwidth of the audio stream. The plain old telephone system (POTS), as we know and love it, uses an 8 kHz audio sampling rate. This is a much lower bandwidth than, say, the desktop, which uses a 22kHz sampling rate. The quality of the audio stream is considerably degraded in the telephony environment, thus making the recognition process more difficult.

The telephony environment can also be quite noisy, and the equipment is quite variable. Users may be calling from their homes, their offices, the mall, the airport, their cars - the possibilites are endless. They may also call from cell phones, speaker phones, and regular phones. Imagine the challenge that is presented to the speech recognition engine when a user calls from the cell phone in her car, driving down the highway with the windows down and the radio blasting!

Another consideration is whether or not to support barge-in. **Barge-in** (also known as **cut-thru**) refers to the ability of a caller to interrupt a prompt as it is playing, either by saying something or by pressing a key on the phone keypad. This is often an important usability feature for expert users looking for a "fast path" or in applications where prompts are necessarily long.

When the caller barges in with speech, it is essential that the prompt is cut off immediately (or, at least, perceived to be immediately by the caller). If there is any noticeable delay (>300 milliseconds) from when the user says something and when the prompt ends, then, quite often, the caller does not think that the system heard what was said, and will most likely repeat what s/he said, and both the caller and the system get into a confusing situation. This is known as the "**Stuttering Effect**."

There is also another phenomenon related to barge-in, and that is called "**Lombard Speech**." Lombard Speech refers to the tendency of people to speak louder in noisy environments, in an attempt to be heard over the noise. Callers barging in tend to speak louder than they need to, which can be problematic in speech recognition systems. Speaking louder doesn't help the speech recognition process. On the contrary, it distorts the voice and hinders the speech recognition process instead.

Conclusions

Speech recognition will revolutionize the way people conduct business over the Web and will, ultimately, differentiate world-class e-businesses. VoiceXML ties speech recognition and telephony together and provides the technology with which businesses can develop and deploy voice-enabled Web solutions TODAY! These solutions can greatly expand the accessibility of Web-based self-service transactions to customers who would otherwise not have access, and, at the same time, leverage a business' existing Web investments. Speech recognition and VoiceXML clearly represent the next wave of the Web.