IBM Pervasive Computing
February 2003

X+V and Speech Considerations

Contents

Executive Summary

XHTML+Voice (X+V) is a markup language for writing multimodal applications, combining visual and voice interfaces. It is based on a number of W3C standards, most notably XHTML and VoiceXML.

Each X+V platform includes technologies for performing speech recognition and speech output. In order to create effective multimodal applications, it helps to understand some of the issues involved in using these voice technologies.

This paper discusses practical grammars, speech recognizer properties, speech output, and TTS customization. It is written for application developers who may be interested in creating X+V enabled multimodal applications.

Practical grammars

In an X+V application, each field or form can specify one or more speech recognition grammars. These grammars determine the utterances (words and phrases) that can be understood by the speech recognition engine. Grammars can be inline (contained within the X+V code) or external (stored in a separate file).

Each X+V platform supports one or more grammar formats, which can include the XML Form of the SRGS (W3C Speech Recognition Grammar Specification), the Augmented BNF (ABNF) Form of the SRGS, and the Java Speech API Grammar Format (JSGF). *Figure 1* is an example of a JSGF grammar:

```
#JSGF V1.0;
grammar command;
public <command> = <action> <object>;
<action> = open | close | delete | move;
<object> = [the | a] (window | file | menu);
```

Figure 1—Example of a JSGF Grammar

When this grammar is active, *open a menu*, *close window*, and *delete the file* are all valid utterances. In fact, this simple grammar enables 36 phrases. Add just one more choice to the <action> rule, and the total number of phrases becomes 45. As grammar complexity increases, the number of phrases grows quickly. The number of phrases can have a significant impact on engine and application accuracy, as well as efficiency.

Engine accuracy

One way to measure accuracy is by how often the speech recognition engine returns the correct phrase, exactly as spoken. This is also known as *engine accuracy*. For example, if a user speaks 50 phrases, and 46 are recognized correctly, the engine accuracy is 92%.

Engine accuracy is directly affected by the number of phrases that are enabled at any given time. In general, the more phrases that are enabled, the more phrases that can be confused with each other, and therefore the lower the engine accuracy.

Some utterances are more easily confused than others. For example, *show games* and *show gains* are likely to be mistaken for each other, whereas *display puzzles* and *graph stocks* are not. Worse yet, homonyms such as *to, two,* and *too* cannot be distinguished at all.

Application accuracy

From the end user's standpoint, the real measure of accuracy is how often the application performs the desired result. This is called the *application accuracy*. For example, if the user says *open my file*, but the engine recognizes this as *open file*, the application might still perform the desired result.

In that case, the engine was not technically accurate, but the application was. If the user speaks 50 phrases, and 46 are recognized correctly, but an additional two phrases are handled correctly by the application, the application accuracy is 48 out of 50, or 96%.

In general, application accuracy can be improved by conducting usability tests to determine the phrases that users are likely to say, and then adding those phrases to the active grammar. For example, in Figure 1, the optional grammar piece *[the | a]* may increase application accuracy. However, the more phrases that are enabled, the more likely the engine will misrecognize one phrase for another. Here are some suggestions:

- Enable phrases that are as long and distinctive-sounding as possible.
- Start with as few phrases as possible, in order to maximize engine accuracy.
- Include alternative phrases that users are likely to say.

- Avoid similar-sounding phrases that have different meanings.

Efficiency

Another consideration when writing grammars is efficiency. Two components of grammar efficiency are *reco time* and *compile time*.

Reco time is the time it takes for the user's utterance to be recognized and converted to text. This depends on many factors, including the speech recognition engine, the spoken utterance, and the complexity of the grammar. The only factor that can be controlled by the grammar author is the complexity of the grammar, so keep in mind that the simpler the grammar, the shorter the reco time.

Compile time is the time it takes for the grammar to be compiled, or converted to a form that is usable by the speech recognition engine. Again, the simpler the grammar, the shorter the compile time. However, X+V platforms may provide a couple of additional features that can be used to shrink compile time:

Pre-compiled grammars are grammars that can be compiled ahead of time, so that X+V applications can reference already-compiled grammars. That avoids the compilation step completely, but may lengthen load time, if the compiled version is much larger than the uncompiled version. Efficiency then becomes more dependent on the speed of the network connection.

Application-specific pronunciation dictionaries. If a speech recognition engine does not have built-in pronunciations for some of the words in a grammar, it may generate pronunciations for those words when the grammar is compiled. X+V platforms may allow grammar authors to specify a pronunciation dictionary, which contains pronunciations for words appearing in the grammar. That keeps pronunciations from having to be generated at compile time. Furthermore, pronunciation dictionaries can be used to customize

pronunciations for unusual words (such as proper names), thereby improving engine accuracy.

Usability considerations

Annotations are symbolic representations that grammars can return when certain words or phrases are recognized. For example, in a flight reservation system, the user may say the name of a city, but the grammar can return the airport abbreviation code. Virtually all grammar formats support some form of annotations. That allows users to say what comes naturally, and have the grammar convert this to the form needed by the X+V application.

Speech recognizer properties

X+V applications can specify a number of speech recognizer properties. These properties affect the performance of the speech recognition engine, and therefore the X+V application. Speech recognizer properties can be set at the form or field level, via the *<property>* element, as shown in *Figure 2*.

```
<vxml:form id="example">
   <property name="confidencelevel" value="0.8"/>
   .
   .
   .
</vxml:form>
```

Figure 2 – Example of the <property> element

Here are descriptions of some of the speech recognizer properties.

confidencelevel

This property sets the *confidence level* (also known as *rejection threshold*) of the speech recognition engine, on a scale from 0.0 to 1.0. Each result from the engine has a confidence score, which indicates how certain the engine is of that result. Results whose confidence

score is at or above the specified confidence level are accepted, but results below that level are rejected.

The higher the confidence level, the less likely the engine is to return an incorrect result. The lower the confidence level, the less likely the engine is to reject a correct result. The appropriate tradeoff depends on the X+V application. If an incorrect result would be disastrous, it is important to specify a high confidence level. Otherwise, the default or a lower confidence should work well under most conditions.

sensitivity

The *sensitivity* of the speech recognition engine is set on a scale from 0.0 to 1.0. The higher the sensitivity, the more sensitive the engine is to quiet input. That is, quiet input is more likely to be recognized correctly. The lower the sensitivity, the less sensitive the engine is to noise. That is, the engine is less likely to misinterpret background noise as valid input.

Again, the appropriate tradeoff depends on the X+V application. If an incorrect result would cause harm, or if users are likely to be in a noisy environment, specify a low sensitivity. Otherwise, the default or a higher sensitivity should work well under most conditions.

speedvsaccuracy

Some speech recognition engines have the ability to trade speed for accuracy, or vice versa. That is, they can operate in a mode that optimizes for speed, in which case accuracy is reduced, or in a mode that optimizes for accuracy, in which case speed is reduced. The *speedvsaccuracy* property allows an X+V application to request optimization for speed, accuracy, or somewhere in between.

The value can range from 0.0 to 1.0. Higher values mean greater accuracy and slower speed; lower values mean lower accuracy and faster speed. High values are good if incorrect results would be difficult to correct

or reverse. Low values are good if it is important for the user to see results quickly. Although higher values mean slower speed, they may actually yield greater efficiency in the long run, since it can be time-consuming for users to reverse the effects of incorrect engine results.

completetimeout

incompletetimeout

maxspeechtimeout

These timeout properties affect when the engine considers an utterance to be finished. Not all speech recognition engines support these properties. See the VoiceXML specification for details.

Speech output

X+V applications can output speech by using either *text-to-speech (TTS)* or *pre-recorded audio*. Speech output can provide:

- Verbal prompts without being limited by screen or window size,
- Additional details about the field requiring input,
- Help when requested, and
- Feedback on data entered via speech recognition.

Several technologies exist for producing TTS output, and X+V platforms may or may not provide a choice. Each TTS technology has its own set of pros and cons, as does recorded audio.

Formant TTS

Formant TTS produces speech output algorithmically from scratch. This has several advantages. First, it does not require a lot of memory or disk space. It has an unlimited vocabulary and will attempt to pronounce

any word. It is also highly configurable and easy-to-understand. However, the resulting speech does not sound very natural.

Concatenative TTS

Concatenative TTS produces speech output by piecing together segments of recorded speech. Each of these segments is typically smaller than a syllable. The advantage is that this sounds more natural than formant. And as with formant, concatenative has an unlimited vocabulary. However, concatenative requires more memory and disk space than formant, and is not as configurable as formant.

Phrase-splicing TTS

Phrase-splicing TTS produces speech output by piecing together entire pre-recorded phrases. The advantage is that this sounds very natural, possibly even indistinguishable from recorded audio. However, phrase-splicing systems must rely on concatenative technology for any words or phrases that were not pre-recorded. Therefore, phrase-splicing works best with a fixed set of phrases, rather than an unlimited vocabulary. If the X+V application's prompts change in the future, new phrases would need to be recorded. Furthermore, phrase-splicing requires even more memory and disk space than concatenative.

Recorded audio

Recorded audio sounds more natural than TTS (especially formant), but must be recorded ahead of time. This means that, as with phrase-splicing TTS, if the X+V application's prompts change in the future, new recordings would need to be made. Additional tradeoffs involved with recorded audio include larger consumption of storage space, and additional file I/O management. On the other hand, pre-recorded files require less memory and CPU processing.

TTS customization

There are several ways that X+V applications may be able to customize TTS, depending on the X+V platform.

Voice characteristics

X+V platforms may allow applications to configure a number of characteristics of the TTS voice. Examples include gender, age, speed, volume, pitch, and pitch fluctuation.

Formant TTS tends to be highly configurable. However, configuring concatenative TTS may cause distortions that diminish pleasantness and intelligibility.

Audio format

X+V platforms may provide a choice of audio formats for TTS. An example of an audio format might be *11k 16-bit mono PCM*, where:

11k is the sample rate, or the number of audio samples per second. Each sample provides a digital snapshot of the sound waves, so the more samples per second, the higher the potential quality. An 11k sample rate actually provides 11,025 samples per second. Other common sample rates include 8k (8,000 samples per second), 16k (16,000 samples per second), and 22k (22,050 samples per second).

16-bit is the number of bits per sample. The number of bits per sample is usually 8 or 16. Again, the more bits per sample, the higher the potential quality.

Mono indicates that there is one audio channel, and is sometimes called “single-channel.” Support for two audio channels would be called “stereo” or “dual-channel.” TTS is normally mono.

PCM stands for pulse code modulation, which is a standard method of encoding digital audio.

Exception dictionary

X+V platforms may support an *exception dictionary*, which is a list of words and their pronunciations. This is useful for words that are not known by TTS system, or that have unusual pronunciations.

The X+V application developer creates an exception dictionary using a text editor or special tool, and the X+V application tells the TTS system (either through a configuration file or an X+V language extension) to load that exception dictionary. Then the TTS system uses the pronunciations in the exception dictionary to override the default pronunciations.

Summary

X+V gives applications a great deal of flexibility regarding the use of speech recognition and speech output. For speech recognition, it is important to create grammars that take into account accuracy and efficiency, and to choose suitable speech recognizer properties. For speech output, it is important to know when to use recorded audio or one of the TTS technologies, and to take advantage of customization features provided by the X+V platform. The result will be X+V applications that can perform well and satisfy end-users.

References

This paper discusses strategy and plans which are subject to change because of IBM business and technical judgments.

All statements regarding IBM future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only.

References in this publication to IBM products or services do not imply that IBM intends to make them available in any other countries.

Performance results obtained in other environments may vary significantly from your results. There is no guarantee that these measurements will be the same on your systems.

Trademarks

IBM, the IBM logo and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

IBM Corporation
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
2-03