

PROJECT JUMBO: IMPLEMENTATION OF A TRANSCRIPTION IM PLUG-IN AS AN ASSISTIVE TECHNOLOGY

Ira R. Forman

formani@us.ibm.com

Allen K. Wilson

wilsona@us.ibm.com



Human Ability & Accessibility Center
11501 Burnet Road
Austin, TX 78759

Abstract. The integration of voice transmission into instant messaging systems disadvantages the deaf. This problem may be mitigated with automated speech recognition (ASR) transcription, which we implemented in IBM's Sametime.

1. Introduction

The integration of voice services using voice-over-IP (VoIP) into instant messaging is almost ubiquitous. As use of this cheap and convenient facility spreads, voice chat interactions will increasingly replace text chat. This change will disadvantage people who are deaf or hard of hearing. The IBM Human Ability & Accessibility Center initiated Project Jumbo to address this problem. Our remedy is to add a speech-to-text capability to augment voice services with transcripts. In particular, Project Jumbo augments IBM Lotus Sametime [1].

2. Assistive technology scenarios

There are two scenarios, described below, that are foremost in shaping our design. At this time (August 2007), we have not yet field tested our software for any of the scenarios; the field test is scheduled for the fourth quarter of 2007.

2.1. Interpreter substitution

Two colleagues, one deaf and one hearing, meet in a conference room. The hearing participant brings a laptop, establishes a chat session with herself, and turns on the microphone. When the hearing participant speaks, the text is displayed in the chat window. The deaf participant can type into the same chat window. In addition, they can draw on the whiteboard, lip read, see facial expressions, and gesture to one another. For the hearing participant, wireless headphones facilitate communication as would a text-to-speech plug-in.

This one computer scenario with its sharing of one keyboard and one screen may be a bit inconvenient. Because our assistive technology is embedded inside an instant messaging sys-




tem, this inconvenience may be eliminated by using as many laptops as there are participants. Instant messaging permits more than two participants.

2.2. Teleconference transcription

Consider a teleconference in which some participants are deaf. Because our software shares the microphone input with the Sametime Voice Suite, a voice over IP (VoIP) facility, the speaking need only turn on the microphone button for text chat voice input. The statements of each speaker are then transcribed and distributed to the text chat windows associated with the teleconference. Deaf participants can provide input by typing into the text chat windows.

3. User interface

Sametime is programmed as a Rich Client Platform (RCP) application [2]. RCP is an open tools platform with which almost any kind of client application can be created by programming the appropriate set of plug-ins. For example, the Eclipse Interactive Development Environment is an RCP application. Our Project Jumbo plug-in extends Sametime so that spoken microphone input can be turned into text for chat input.

Figure 1 depicts the initial Sametime chat window for interaction between the authors (Allen's side). To begin, Allen depresses the microphone button  on the lower right side of the window. The icon changes to  and Allen may now speak. There is no explicit *send* command: if you pause for more than p milliseconds or speak more than w words, your utterance is sent. Both p and w are preference parameters that are initially set to 1500 milliseconds and 12 words. If the button  is depressed, the microphone is turned off. This implies that the plug-in operates hand free once the microphone is turned on.

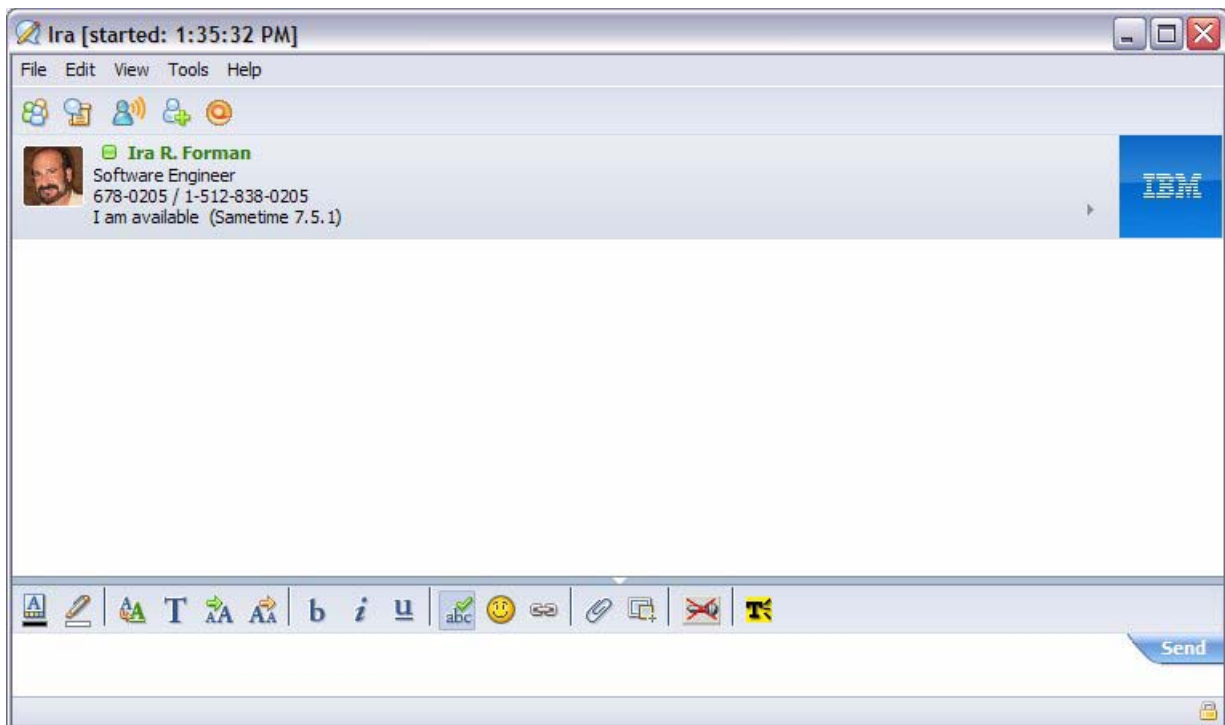


Figure 1. Example of a Sametime chat window.



A user may have multiple chat windows open simultaneously. Each chat window has a microphone button . When one of those buttons is depressed, that chat window has the speech focus. The speech focus stays with that chat window until the microphone is turned off. That is, speech focus does not follow the mouse cursor. Speech focus may be moved among chat windows by turning on the microphone button in another chat window (which transfers focus). Of course, closing the chat window with the speech focus turns off the microphone. (Note that the  icon in Figure 1 is the mute button for an IBM-internal TTS plug-in, which is implemented by the Embedded ViaVoice [3] team).


Figure 2 depicts a conversation between the authors. The upper frame contains the transcript; the lower frame is the typing area. Spoken text is transmitted directly to the transcript frame (consequently, the typing area is empty). The spoken text icon  indicates that the line is spoken rather than typed. The nature of the errors made by an ASR engine differs from typed errors. Consequently, an indicator for spoken text is important. This icon is displayed only once at the beginning of each contiguous set of lines spoken by a user.



Figure 2. Example of a chat in which one participant is using the microphone as the input device.

Figure 3 depicts an active Voice Suite window. Figure 3 shows a repeat of the conversation in Figure 2. The main difference between the two is that when using the Voice Suite the participants have an audio connection. The left column displays the participants as well as shared documents. The column on the right is a text chat window displaying the transcript from the ASR engine. (Both Allen and Ira are speaking in Figure 3; the difference with Figure 2 in the display of the transcript is a software defect that will be resolved shortly.)

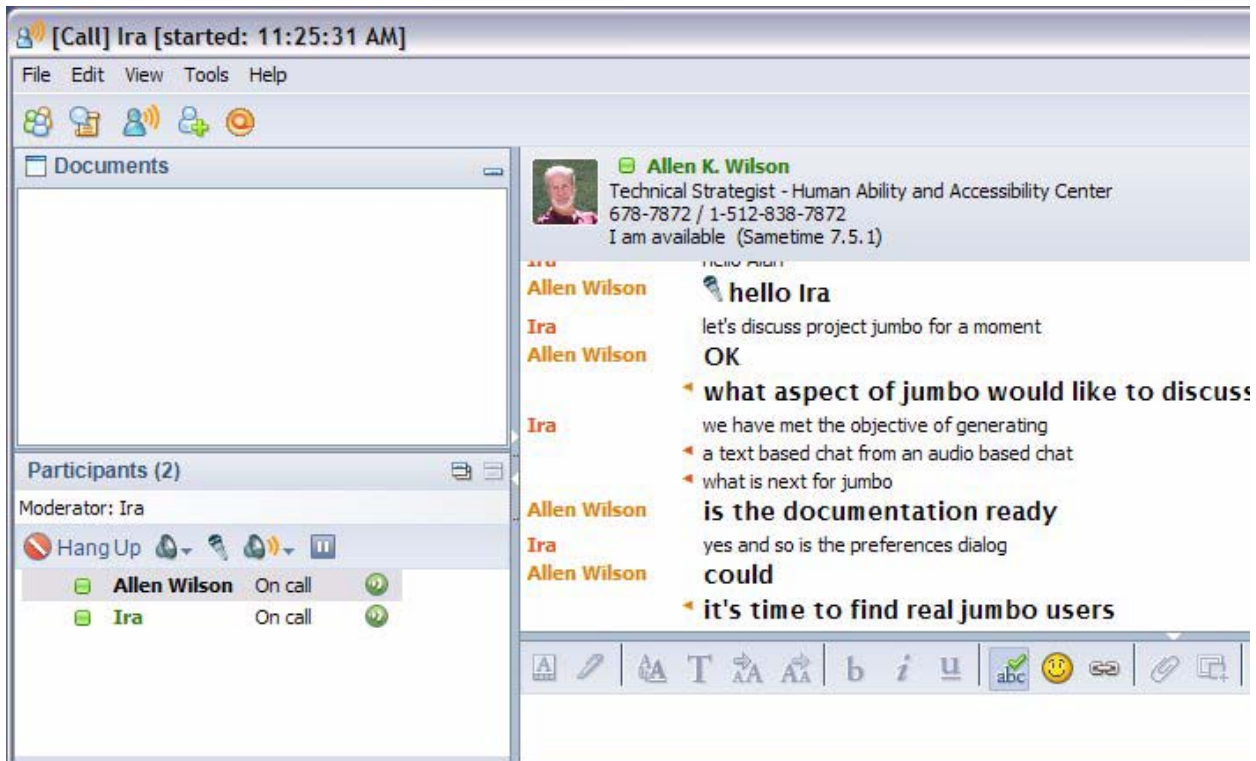


Figure 3. Example of a Voice Suite chat window.

4. Why we expect this to succeed

The limitations of the currently available ASR engines are a prime consideration in our design. Attaining the best performance requires each speaker to have a personal microphone of good quality and a personal ASR engine that the speaker has trained. That is, our design is not addressing the problems of creating a meeting transcription where there are open shared microphones and ASR engine must either work in a speaker independent mode or identify the speaker.

Even when properly trained, current, commercially-available ASR engines exhibit a word error rate of about 10% for large vocabulary speech recognition. Such a word error rate appears to be disappointing as most people who have tried dictation products will tell you. Properly composing a written document requires much revision. Further, using the mouse to relocate the insertion point is more effective than oral commands. Consequently, dictation systems are not generally more effective for composing written documents than typing alone (the low penetration of dictation systems into the personal computer market is sufficient evidence for this statement).

This raises a serious question: why should we expect Project Jumbo to succeed when ASR is so disappointing in the apparently similar application, dictation?

First, the plug-in is operating in an interactive context. Many errors are not corrected by the speaker because the “listener” (the other users who are reading the ASR engine output) can easily determine what is meant. For example, if one speaks “I think you need to fix that” but the ASR emits “I fink you need to fix that,” the “listener” should be able to correct the error. (A good speaker should take into account the language skills of the listeners in such cases.)

Second, the errors made by the ASR engine are not uniformly distributed; a significant number come in clumps. In such cases, the speaker merely repeats the statement (with better articulation). This is analogous to all of our experience in using cell phones; some statements get horribly garbled. In the case of cell phones, the listener typically asks the speaker to repeat him-

self. Our case is different in that the speaker can see the output of the ASR engine slightly before the “listener.” We are experimenting with protocols for error correction. For example, if the transcription of a statement is unsatisfactory, the speaker says “correction,” pauses (to get the word transmitted), and repeats the statement.

Third, typing is not disabled and can be used profitably to augment speech input. Typically, this happens when some intricate technical term needs to be communicated (speaking a long URL is not very enjoyable). Another typical case occurs when the speaker realizes that the “listener” does not have the language skills to correct the errors of the ASR engine.

For these reasons, automated speech recognition even with a 10% word error rate is expected to be effective in the interactive contexts addressed Project Jumbo while the 10% word error rate makes archival use of the transcripts problematic.

5. Future work

With the basic software capability implemented, our next tasks will be to understand its use in its intended social context. There are number of tasks still to be performed.

- **Field testing:** All of the scenarios of Section 2 must be tried. Protocols of interaction must be devised and the software modified to ensure the successful interactions among the people involved in the scenario.
- **Answer the unanswered questions:** There are many questions that need answering. Foremost among these is: What word error rate is too high to successfully use ASR in a particular scenario?
- **Other ASR engines:** The plug-in prototype employs ViaVoice 10.5 [4]. The plug-in should be extended to use other popular ASR engines. If a user happens to have gone to the trouble to train an ASR engine, why should we, the software developers, force the user to train another? In addition, we will consider integration of forthcoming IBM engines [5].

6. Conclusion

Project Jumbo has really just begun. The efficacy of current ASR engines to provide a usable transcript for the deaf and hard of hearing has yet to proven. Be that as it may, our preliminary results are very encouraging.

Acknowledgements

Project Jumbo could not have gotten this far without the generous help of Thomas Brunet, Ann Ford, David Jaramillo, Paul Luther, Ann Marie Maynard, Mike Strack, Kavitha Teegala, Giant Tu, and Arro Walter. For their insightful comments on this paper, we are indebted to Thomas Brunet, Barry Feigenbaum, Andi Snow-Weaver, and George Stark. We would also like to thank Dimitri Kanevsky, Sara Basson, Edward Epstein, and Peter Fairweather for their patent [6], which obviated any intellectual property issues.

References

1. www-142.ibm.com/software/sw-lotus/products/product3.nsf/wdocs/st75home
2. www.eclipse.org/articles/Article-RCP-1/tutorial1.html
3. www-306.ibm.com/software/pervasive/embedded_viavoice
4. <http://www.nuance.com/viavoice>
5. Ramabhadran, B., “The IBM 2006 Speech Transcription System for European Parliamentary Speeches,” www.tc-star.org/pubblicazioni/scientific_publications/IBM/tcstareval06.pdf, 2007.
6. Kanevsky, D. et al., “System and Method for Teleconferencing with Deaf or Hearing Impaired,” US Patent 6,618,704, 2003.