A Java Speech Implementation of the Mini Mental Status Exam

Stephen S. Wang, Ph.D.* and Justin Starren, MD, Ph.D.†
Departments of Medical Informatics and Radiology†
Columbia University, New York, New York

The Folstein Mini Mental Status Exam (MMSE) is a simple, widely used, verbally administered test to assess cognitive function. The Java Speech Application Programming Interface (JSAPI) is a new, cross-platform interface for both speech recognition and speech synthesis in the Java environment. To evaluate the suitability of the JSAPI for interactive, patient interview applications, a JSAPI implementation of the MMSE was developed. The MMSE contains questions that vary in structure in order to assess different cognitive functions. This question variability provided an excellent test-bed to evaluate the strengths and weaknesses of JSAPI. The application is based on Java platform 2 and a JSAPI interface to the IBM ViaVoice recognition engine. Design and implementations issues are discussed. Preliminary usability studies demonstrate that an automated MMSE maybe a useful screening tool for cognitive disorders and changes.

INTRODUCTION

Keyboard and video display have been the principal means of entering and retrieving data for the past 20 years. As computers become more and more pervasive in many areas of modern health care, it becomes increasingly important to develop more intuitive user interfaces to allow easy human computer interactions. In addition of being ease of use, speech technology is almost imperative in some situations where people's hands are otherwise occupied. Speech technology has been integrated into applications for health care including decision support systems, voice-activated radiology report generators, and a voice-enabled medical reporting system.

Widespread integration of speech recognition into medical applications has been limited by the complexity and proprietary nature of previous hardware and software solutions. Recent advances in computer hardware and software have made speech recognition become more and more available in real applications.

The Java Speech API (JSAPI) defines a software interface which allows developers to take advantage of speech technology from within Java programs. It provides programmers a uniform, cross-platform speech API to develop more sophisticated and natural user interfaces. JSAPI supports speech synthesis, grammar driven command-and-control recognition, and free text recognition for dictation systems. The JSAPI is an interface to the speech recognition components. It does not actually perform speech recognition. Third party speech recognition software is still required. For this current implementation, the IBM JSAPI implementation utilizing the ViaVoice recognition engine was selected.

To evaluate the capabilities of the JSAPI, a speech-enabled version of the Mini Mental Status Exam (MMSE) has been implemented. The Folstein's MMSE has 11 questions to test patients' cognitive functions and it has been widely used. The MMSE was selected because it provided a variety of speech interface challenges. Each in the 11 questions of the exam tests different cognitive function and each requires different user interface characteristics.

APPLICATION DESIGN

A major design challenge is the smooth integration of graphical and speech subsystems. Figure 1 shows the schematic design of the application. The application logic controls the program flow. The GUI provides a conventional visual display and allows for a mouse-
base interaction with the system. The JSAPI component handles all the speech-related activities: the synthesizer takes text input and converts it into speech; the recognizer accepts voice input and outputs tags for use by the application logic. The timer component sets time limits for each question. The scoring component calculates the scores based on individual answers. The voice output device is headphones or speakers. The voice input device is a microphone.

IMPLEMENTATION

User Interface Issues
Most of the MMSE questions could be implemented without alteration. For example, questions regarding date and place are unaltered. Questions involving orientation or computation, in addition to being spoken by the speech output, are also shown on the screen to provide patients with additional cues (e.g., the serial sevens and naming questions). For other questions, displaying the questions on screen would invalidate the questions. For example, the question asking the patient to memorize and repeat three objects would be invalidated if the objects were shown on the screen. In this latter case, only general cues are provided. For some questions, intervention cues are provided. For example, the serial sevens question asks the patient to count backwards from one hundred by 7 and to stop after five answers. Since the speech recognition is never 100% accurate, showing what the computer has recognized provides a way of confirming the patient's progress (Figure 2).

Some questions required extensive alterations. These were typically questions that require human observation. In these cases, analogous tasks were created that are easier for the computer to monitor.

The verbal command question tests the patient's ability to perform a sequence of multiple actions. The original question asks the patient to "take a piece of paper in his/her right hand, fold it in half, and put it on the floor". Without advanced machine vision, it is impossible to implement this question with the current system. We modified the question but the new one conforms to the goal of the original. The new question asks the patient first to click on the big button, then the small button, and finally say "It is done" (Figure 3).

Another example of a question that required modification was the test of the patient's reading ability. In the original question, the patient was handed a piece of paper with the words "Close your eyes" on it. Since the computer system does not have vision recognition capability, we modified the question to instruct the patient to click a red circle on the screen.

Another question requiring alterations asks the patient to write a sentence. The implementation here is to present the patient a writing dialog box and ask him/her to type in a sentence.
Grammar Construction

The speech recognition for this application utilized rule-based grammar. Rule-based recognition is well suited to cases where the number of inputs is limited and the computer must respond to specific inputs. Compared with free dictation recognition, grammar-based recognition in general provides higher accuracy of recognition. The grammar construction for JSAPI utilizes the Java Speech Grammar Format (JSGF) 1.0. For most of the questions, the recognition grammars can be preprogrammed. For the date question, however, since the date changes everyday, the date grammar is dynamically generated when the application is started. By utilizing the Java Calendar class, the information on year, month, day of month, day of week is readily available. Season information can be calculated by the month information.

Each grammar rule is one line of a text file that has a grammar name and is tagged. When a grammar rule is recognized, the associated tag is returned to the application. New grammars can be generated by combining existing grammars. These combinations can generate unexpected behavior when the Java speech grammar recognizer tries to recognize speech input. For example, there are grammar rules for month named <p_month> and for day of month <p_day>. Also there is a combined grammar rule named <p_month_day> which has both month and day of month information. When the patient tries to answer the month and the day of month in one continuous speech sentence, e.g., February 24, the recognizer will return the tags for all the recognized grammar rules in an array. In this case, the first recognized rule is <p_month_day>, <p_month> and <p_day> will be also in the array. The tag order does not follow any obvious pattern. (Table 1). In another example, where there are tags of <p_season>, <p_dayofweek>, and <p_season_dayofweek>, the order of recognition when the patient says "Winter Wednesday" is first <p_season>, second <p_dayofweek> and third <p_season_dayofweek>. As a result, the entire array must be evaluated at each iteration.

<table>
<thead>
<tr>
<th>Patient Speaks</th>
<th>Order of Recognized Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 24</td>
<td>(1) &lt;p_month_day&gt;</td>
</tr>
<tr>
<td></td>
<td>(2) &lt;p_month&gt;</td>
</tr>
<tr>
<td></td>
<td>(3) &lt;p_day&gt;</td>
</tr>
<tr>
<td>Winter Wednesday</td>
<td>(1) &lt;p_season&gt;</td>
</tr>
<tr>
<td></td>
<td>(2) &lt;p_dayofweek&gt;</td>
</tr>
<tr>
<td></td>
<td>(3) &lt;p_season_dayofweek&gt;</td>
</tr>
</tbody>
</table>

Table 1. Recognition of java speech grammar.

Text-to-Speech

To interact with the user the program utilizes a speech synthesis. Effective use of speech synthesizers often requires text markup to ensure correct pronunciation and phrasing, emphasis and speaking rate. The Java speech synthesizer utilizes Java Speech Markup Language 1.0 (JSML). Several tags are used in the application. <EMPH> </EMPH> are placed in several places when the emphasis of certain words or phrases are needed. <PROS RATE="-20"></PROS> are used when slower speeding of speaking is needed to make it more understandable.

Figure 4. Screen shot of the score of drawing question. Left panel shows the original pentagon drawing and right panel shows the pentagons drawn by a user.
**Timer Component**

Two timers used in the system are implemented as Java beans (Symantec Visual Café). One timer serves as the master control for each question, i.e., the timer sets a time limit for each question. One minute time limit is set for each question except the drawing pentagon one in which two minutes time limit is set. The second timer is a reminder timer. It informs the user what the computer has understood and prompts the patient what to do next. The second timer is triggered every 21 seconds for the first 10 questions and every 45 seconds for the drawing pentagon question.

When the patient answers correctly, the timer must be stopped and restarted again for the next question. When the patient fails to answer the question, besides stopping and restarting, the timer needs to trigger the voice output. When the patient pauses on the question for too long, the timer needs to remind the patient that he/she should try to answer the question. The timer is implemented as a Java bean thread and it demonstrates the integration of JSAPI with multi-threaded Java.

The timer plays an extremely important role in the whole system behavior. The system can not recognize the patient's voice with 100% accuracy. When the patient's voice answers are not recognized, the system cannot distinguish mis-recognition from incorrect answers. The timer implementation elegantly compensates for this ambiguity.

**Scoring**

In the original Folstein's MMSE, each correctly answered sub-question represented one point with a maximum score of 30. In our implementation, scoring utilizes a boolean variable for each sub-question. If a question is answered correctly, the boolean variable is set to be true. For some questions, the system must integrate input from both the screen and the speech subsystems. For example, in the verbal command question, the system needs to score on both clicking onscreen buttons and speech input (Figure 3). No natural language processor capability is implemented in the system. We are exploring the possibility of feeding the patient's input into an external syntactic parser. At current stage, the score for writing a sentence must be done by a human reviewer.

Scoring of the pentagon drawing question presented a special challenge. The two pentagons drawn by the user should only have one vertex from each pentagon residing in the other pentagon. By using the Polygon class provided by the Java 2 platform API, we have been able to implement the scoring mechanism to solve this seemingly complicated image recognition problem. Using the `inside` method of the Polygon class, each vertex of one pentagon can be tested to see if it is within the boundary of the other pentagon. At the end of the exam, a total score is presented to the user.

**USABILITY**

Usability testing was performed on 5 subjects (subjects are graduate students at the Department of Medical Informatics of Columbia University). Overall users were satisfied with the system. Users were a bit nervous in the beginning of the test. But as the test proceeded, they became more comfortable. The variations of scores were predominantly due to failure of the recognition engine and misinterpretation of the instructions by the subjects. Recognition errors generated significant frustration on the users. Also, users sometimes were tentative in answering questions since they were not sure what exact answers that the computer system was looking for. One user complained that the computer voice was "less human". Another stated that the instruction given by the system in the beginning of the exam should be more thorough. Users also suggested rephrasing of some questions for better understanding. For comparison, standard human-administrated MMSE was also performed on these subjects. The average score of the exam administered by a human is 26 and the average score of the exam administered by the computer system is 24.8.

**DISCUSSION**

Although speech technology can be traced back to the early stage of computer development, it is still at its infancy stage. Compared to other user interface methods, e.g., keyboard and mouse, it is still far less accepted. But it promises a more naturally user interface and more intuitive human-computer interaction.

Java speech API, a recent member of the rapidly increasing Java package family, provides the developers' community, for the first time, a cross-platform, vendor-independent API. Developers can choose the speech products from various vendors to best meet their needs. Java applications with a more human appearance can be developed.

Java speech is still a cutting edge technology and needs time to mature. The JSAPI specification advanced from version 0.7 to version 1.0 in the process of our development. The IBM
implementation of JSAPI is not a full implementation and is still at a beta stage. This led to a few problems during our development. For example, the JSAPI implementation does not handle multiple grammar imports or nested grammar imports.

Historically, the traditional MMSE has been a very efficient way of assessing patients' mental status. It takes minutes to finish and does not require any special devices. Although we have automated the entire MMSE, the system will not replace the human being intervention on every case. Our primary goal was to test the JSAPI capabilities and usefulness in medical applications rather than to meet specific clinical needs. For significantly impaired patients, interaction entirely with the computer is probably unpractical. However, this system still has potential clinical use. For instance, it could be used as a routine screening tool for cognitive disorders and changes.

Even the reported accuracy could be up to 98% for the speech recognition engine, the functional accuracy is normally much lower. In this study, the recognition varied from above 90% to below 50% depending on the speakers and questions. Given the rapid improvement of the speech technology, we anticipate that an accurate speaker-independent speech recognition is not far off.

Java platform and Java Speech API provide new technologies to developers to implement innovative applications. The implementation of the mini mental status exam has shown us an example what can be achieved by using these technologies.

Acknowledgement

The authors thank Michael Charney for technical discussion, Drs. Jinsong Cai, Ying Chen, Charles Mun, Michelle Robinson, Irina Shablinsky for testing the system. This project is supported by the National Library of Medicine post-doctoral training grant - #LM-07089, and the New York State Science and Technology Foundation.

References