An Intelligent Interface for Keyboard and Mouse Control
– Providing Full Access to PC Functionality via Speech

Bill Manaris
Computer Science Department
College of Charleston
Charleston, SC 29424
manaris@cs.cofc.edu

Renée McCauley
Computer Science Department
College of Charleston
Charleston, SC 29424
mccauley@cs.cofc.edu

Valanne MacGyvers
Psychology Department
University of Louisiana
Lafayette, LA 70504
macgyver@louisiana.edu

ABSTRACT
SUITEKeys is a speech user interface for motor-disabled computer users. This interface provides access to all available functionality of a computer by modeling interaction at the physical keyboard and mouse level. SUITEKeys is currently implemented for MS Windows platforms. Its architecture integrates a continuous, speaker-independent recognition engine with natural language processing components. It makes extensive use of dialog management to improve recognition accuracy. *SUITEKeys* extends the speech-keyboard metaphor through functionality visible on its graphical user interface, and accessible through speech. Experimental results indicate that speech input for alphanumeric data entry is a much more effective modality than existing alternatives for the target user group. Such alternatives include miniaturized keyboards, stylus "soft" keyboards, and handwriting recognition software.

Keywords
Speech recognition, natural language processing, intelligent user interfaces, accessibility.

INTRODUCTION
Studies indicate that speech interaction with a virtual keyboard and mouse is a very effective input modality for motor-control challenged users, in terms of data entry, task completion, and error rates [1, 2, 3]. These results apply to both permanent and temporary (task-induced) motor disabilities. Examples of the latter include users entering arbitrary alphanumeric data in mobile computing devices and hands-busy environments. Speech interaction with a virtual keyboard and mouse has been shown to be far better than alternative modalities such as mouthstick, handstick,[1] miniaturized keyboards, stylus "soft" keyboards, and handwriting recognition software [3].

This paper presents *SUITEKeys* 1.0—a speech user interface for providing access to a virtual keyboard and mouse. *SUITEKeys* integrates state-of-the-art components for speech recognition, natural language processing, and dialog management, to provide a speaker-independent, continuous-speech interface. It is implemented on MS Windows platforms (95, 98, NT, 2000, and potentially Windows CE—for palmtop PCs).

This speech-interface concept provides motor-disabled users with universal access to any device that requires/supports alphanumeric data entry, such as palmtop PCs, cellular phones, and camcorders. It may also be used by able-bodied users with temporary, task-induced motor disabilities, such as users entering arbitrary (non-word) alphanumeric data in mobile computing and/or hands-busy environments. Specifically, as the physical dimensions of traditional input devices shrink, the motor skills of the user become less effective, almost inadequate. For instance, studies show that users may experience severe reduction in data entry speed (words per minute) when switching from a regular QWERTY keyboard to a mobile-device keyboard alternative. Specifically, these studies report a 75% reduction on a stylus-based "soft" keyboard (PalmPilot) and a 60% reduction on a telephone keypad [4, 5].

SUITEKeys allows users to transcribe sequences of keystrokes and mouse actions using spoken language. It assumes that the user speaks English and has minimal or no speech impediments. It models an 84-key keyboard (no numeric pad) and two-button mouse functionality.[2] Other applications, including the operating system, treat the generated keyboard and mouse events as having originated from the corresponding physical devices.

Other speech applications, such as NaturallySpeaking and Microsoft Voice, do not provide access to all the functionality available through the physical keyboard and mouse [6, 7]. This is because they are designed for higher-level tasks, such as Dictation and Command & Control. They allow input of keystrokes either by requiring a fixed keyword as prefix (e.g., "Press a"), or by entering a spell

[1] Mouthstick and handstick are devices used by motor-disabled users to manipulate QWERTY keyboards.

[2] SUITEKeys models only the left keys for ALT, CONTROL, and SHIFT.

mode. Moreover, they do not necessarily model all 84 keys. Finally, they send recognized keystrokes directly to the active window, bypassing the operating system. This prohibits the operating system and other applications from attaching arbitrary semantics to sequences of keystrokes (e.g., ALT-CONTROL-DEL), and intercepting them.

OVERVIEW

SUITEKeys was originally developed as a testbed application for the SUITE speech-understanding-interface architecture at the University of Louisiana at Lafayette [8]. It has been finalized at the College of Charleston. It is distributed as freeware for non-profit use. An early prototype of the system, the underlying architecture, and theoretical foundations for this work were presented in ASSETS-98, FLAIRS-99 and elsewhere [1, 2, 9].

SUITEKeys has been implemented using MS Visual Studio, SAPI, and Lex and YACC. Minimum system requirements include: Pentium 120 processor (preferred Pentium 200 and up); 16MB RAM for Windows 95 & 98, 24MB RAM for Windows NT, 64MB RAM for Windows 2000; 18MB disk space; sound card & sound card device driver supported in Windows; microphone (preferred noise-reducing, close-talk headset), and 16 kHz/16 bit or 8kHz/16 bit sampling rate for input stream.

SUITEKeys is available at www.cs.cofc.edu/~manaris/SUITEKeys/ .

Domain Limitations

Although the system incorporates a continuous, speaker-independent engine, this capability could not be exploited in all cases. This is due to speech ambiguities in the linguistic domain which normally confuse even human listeners. For instance, the linguistic domain includes several near-homophone letters, such as "b" and "p", or "d" and "t".[3] For this reason, a few tasks have been modeled only through discrete speech. This requires that the user pause between words. Although this may make interaction less natural in some cases, it improves understanding accuracy considerably.[4]

Specifically, we use discrete speech mainly when transcribing regular letters (e.g., "a", "b"). We also use it in a few other cases—where our usability studies showed that users used it anyway—such as entering function keys (e.g., "Function Eleven", "Function Twelve") and certain special keys (e.g., "Insert", "Home", "Print Screen"). We use continuous speech in all other cases, such as entering letters using the military alphabet, numbers, repeatable keys (e.g., "Up Arrow", "Tab", "Back Space"), mouse commands, and system commands.

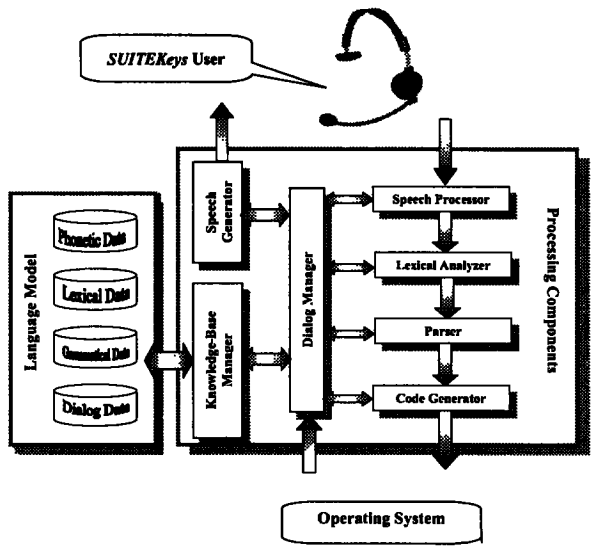

Fig. 1. *SUITEKeys* Architecture.

SYSTEM ARCHITECTURE

The architecture of *SUITEKeys* integrates speech recognition and natural language processing components (see Figure 1). It processes input that originates as speech events and converts it to operating system keyboard and mouse events. The complete system runs on top of the operating system like any other application. Other applications are unaware that keyboard/mouse events originated as speech input.

The architecture is subdivided into (a) the language model and (b) language processing components. Speech events are processed in a pipelined fashion to enable real-time response [2, 8]. Processing is performed by the following components:

Dialog Management

The *SUITEKeys* architecture incorporates a stack-based dialog manager. This subsystem utilizes a dialog grammar containing dialog states and actions. Each top-level state corresponds to a specific language model (e.g., lexicon, grammar). The overall linguistic domain has been divided into thirteen such states. The dialog manager loads, in real time, the appropriate dialog state based on the current state of the interaction. Given the ambiguity of the linguistic domain, this allows processing components to focus on specific subsets of the overall domain, and thus improve recognition accuracy and performance.

Speech Processing

SUITEKeys evolved through several architectures following the rapid developments in speech recognition within the last six years. Currently, it implements the Microsoft Speech API (SAPI). The latest version of *SUITEKeys* comes

[3] Hence the invention of the military alphabet.

[4] We use the term "understanding" as opposed to "recognition," since the system, in addition to translating an utterance to its ASCII representation (recognition), it also carries out the intended meaning (understanding), such as type a letter, move the mouse, and switch speakers.

bundled with the MS Speech Recognition engine (freeware), but could also work with other SAPI-compliant engines, such as Dragon Systems NaturallySpeaking [6].

Natural Language Processing

The architecture incorporates a left-to-right, top-down, non-deterministic parser. This parser supports multiple parse trees, thus providing for ambiguity handling at the semantic level. It incorporates semantic actions for constructing semantic interpretations of user input.

Code Generator

This module converts semantic interpretations—the output of the parser—to low-level code understood by the operating system.

Other Components

Other components of the *SUITEKeys* architecture include a knowledge-base manager, speech generator, lexical analyzer, and pragmatic analyzer. The application domain (as implemented in the final release of *SUITEKeys*) does not have significant lexical, pragmatic, and speech generation requirements. A detailed discussion of these components appears in [2].

GRAPHICAL USER INTERFACE

Although *SUITEKeys* is primarily a speech user interface, it includes a graphical user interface for visibility and feedback purposes. Specifically, the user interface identifies available functionality and how to access it (visibility); it also provides information about the effects of user actions (feedback).

The *SUITEKeys* graphical user interface has two views: maximized and minimized.

Maximized View

This view is designed to provide maximum feedback to the user. As shown in Figure 2, this view allows the user to focus on *SUITEKeys* and its operation. This view provides several forms of feedback (left-to-right, top-to-bottom):

- **Command History:** a list of recognized commands in reverse chronological order. Unrecognized commands are denoted by "???".
- **Speech Processing:** superimposed animations provide feedback for two independent yet related events: voice input level corresponds to raising and dropping rings in green, yellow, and red (Figure 3); speech engine processing corresponds to a revolving "SK" (Figure 4).
- **Toggle Keys:** on/off indicators for CAPS LOCK and SCROLL LOCK.
- **Generated Keystrokes:** a keyboard that visually "echoes" generated keystrokes.

Additionally, this view identifies all system commands through its menu structure. Since some system commands disconnect the speech engine from the microphone (e.g., Microphone Calibration), such commands are not

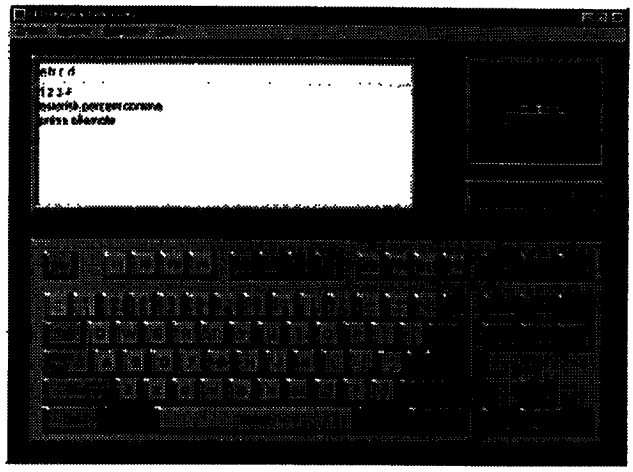

Fig. 2: *SUITEKeys* maximized view.

Fig. 3: Input level.

Fig. 4: Input processing.

Fig. 5: *SUITEKeys* minimized view.

accessible through speech. This is to prevent users from entering system states from which they may not be able to exit.[5]

Minimized View

Effective user interfaces should be invisible to the user. As Mark Weiser states,

"[a] good tool is an invisible tool. By invisible, I mean that the tool does not intrude on your consciousness; you focus on the task, not the tool. Eyeglasses are a good tool — you look at the world, not the eyeglasses." [10].

For this reason, *SUITEKeys* provides a minimized view. This view is intended for when the user wants to concentrate on the task at hand, i.e., controlling the PC, as opposed to *SUITEKeys* itself (see Figure 5). This view always stays on top and provides minimal feedback, namely the last-recognized input, which modifier/toggle keys are pressed, the volume indicator, and the speech engine processing status.

[5] These commands are only accessible to users with some motor control via the physical keyboard/mouse interface.

KEYBOARD SPEECH COMMANDS

SUITEKeys allows users to input sequences of keystrokes using spoken language, much as if they were transcribing to someone how to perform the same actions with a physical keyboard. Specifically, keys may be tapped, pressed, or released. Typing errors may be handled similarly to physical keyboard-entry errors, i.e., via the "Back Space" key.

In the following sections, unless otherwise specified, users may use continuous speech.

Typing Alphanumeric Keys

Given the difficulty of recognizing certain letters of the alphabet, the system supports both regular and military pronunciations.

Saying the name of a letter (e.g., "a", "b") sends the corresponding letter to the active window. To improve accuracy, only discrete speech may be used for sequences of alphanumeric keystrokes.

Saying the military equivalent of a letter (e.g., "alpha", "bravo") sends the corresponding letter (e.g., "a", "b") to the active window.

Saying a number (e.g., "one", "two") sends the corresponding number to the active window.

Word Auto-Completion

Studies show that word prediction techniques improve user performance considerably. This is accomplished by giving the user a choice of the most-likely words, based on previous input. This reduces the overall number of keystrokes a user has to enter by as much as 44% [11].

SUITEKeys incorporates a word auto-completion mechanism. A word prediction box is displayed when the user begins typing a new word. This box is attached to bottom of the mouse pointer for easy positioning. Word auto-completion allows the user to reduce the number of "keystrokes" needed to enter a previously used word. Instead of typing the entire word, the user types enough letters to make it appear in the prediction box. Saying "Select" types the rest of the word. Words are maintained using a most-recently-used scheme (identical to the paging scheme in operating system literature).

Typing Modifier Keys

Modifier keys are held pressed while typing alphanumeric keys to provide semantic variations.

Saying "Press" and a modifier key (i.e., "Alt", "Control", "Shift") holds the key pressed. Saying "Release" and a modifier key, releases the key. The graphical user interface provides feedback as to which modifier keys are pressed at any time.

Typing Toggle Keys

Saying "Caps Lock", or "Scroll Lock" toggles the corresponding key. The graphical user interface identifies toggle key status (on/off).

Typing Other Keys

Saying the name of other keys (e.g., "Function Twelve", "Escape", "Space", "Ampersand", "Tab", "Left Arrow") types the corresponding key.

MOUSE SPEECH COMMANDS

SUITEKeys allows users to manipulate a two-button mouse using spoken language, much as if they were transcribing to someone how to perform the same actions with a physical mouse. Specifically, the mouse may be moved in different directions; its buttons may be clicked, double-clicked, pressed, or released.

SUITEKeys provides three different techniques for setting the position of the mouse pointer.

Continuous Motion

Users may smoothly move the mouse pointer towards a specific direction. Movement continues until a command is spoken (e.g., "Stop"), or until the pointer reaches a screen border. This is intended for rough positioning of the pointer, when the target is far from the original position.

For example, saying "move mouse down" ... "stop", will move the pointer downward some distance. Saying "move mouse two o' clock" ... "stop" will move the pointer towards two o'clock.[6]

Relative Positioning

Users may reposition the mouse pointer a number of units from its original position. This is intended for fine-tuning the mouse position, when the mouse pointer is near the target.

For example, saying "move mouse down three" will reposition the pointer 3 units below its original position. Saying "move mouse three o'clock forty" will reposition the pointer 40 units to the right.

SUITEKeys subdivides the screen into 100 units—"zero" to "ninety-nine"—on both horizontal and vertical axes. This makes relative positioning independent of screen resolution.

Absolute Positioning

Users may reposition the mouse pointer to a specific screen location. This is intended for targets whose approximate screen location is known.

For example, saying "set mouse position three five" will place the pointer 3 units away from the left screen border and 5 units away from the top. In other words, "zero zero" corresponds to the top-left corner and "ninety-nine ninety-nine" to the bottom-right corner.

Controlling Mouse Buttons

Mouse buttons may be controlled similarly to keyboard keys. For example, saying "Click left button" or "Double-click right button" will perform the corresponding actions.

---

[6] Twelve o'clock is the same as up, three o'clock is right, six o'clock is down, etc.

Dragging is achieved by pressing the left mouse button (e.g., "Press left button") and moving the mouse. Dropping is achieved by releasing the left mouse button (e.g., "Release left button").

OTHER FUNCTIONALITY

SUITEKeys provides additional functionality through the menu structure in maximized view. This functionality is also accessible through speech, as long as it does not disconnect the microphone from the speech engine.

Context-Sensitive Help—"What Can I Say"

Studies show that *there are many names possible for any object, many ways to say the same thing about it, and many different things to say. Any one person thinks of only one or a few of the possibilities.* [12]

For this reason, every dialog state includes the command "What Can I Say". This command brings up a box with available commands in this context. This box also appears after a number of unrecognized inputs (currently 4). For example, Figure 6 shows the corresponding box for the Preferences dialog state. Once a correct command is spoken, the box disappears.

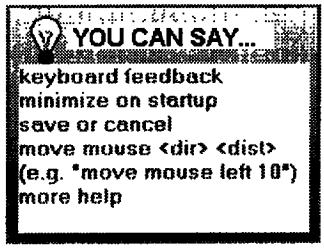

Fig. 6: Context-sensitive help.

Progressive Disclosure—"More Help"

SUITEKeys incorporates a two-step, progressive-disclosure mechanism for help. Specifically, in most dialog states, one of the commands displayed as a result of "What Can I Say" is "More Help". When spoken, this command brings up the corresponding topic in the Help file.

Quick Training

SUITEKeys incorporates training "cards" to teach system essentials to the user. These cards are displayed automatically for every new speaker. Each card corresponds to a training exercise on a subset of *SUITEKeys* functionality. Each training exercise is associated with a very small linguistic model. Training cards are designed so that a user may proceed to the next lesson only by mastering the concept being presented (e.g., mouse control).

USABILITY STUDIES

We have conducted several usability studies related to *SUITEKeys*. Most of them were of formative nature; they provided feedback and direction during development [1, 2, 3]. These results indicate that interaction with a speech-accessible keyboard and mouse is a very effective input modality, in terms of user data entry, task completion, and error rates. Specifically, users performed much better using a Wizard-of-Oz model of *SUITEKeys* as opposed to handstick. Handstick is a good approximation of alternate input modalities such as mouthstick, miniaturized keyboards, stylus "soft" keyboards, and handwriting recognition software. Specifically, the average results on our particular benchmark task were as follows:

- Time to completion: 201.5 secs for speech; 321.1 secs for handstick.
- Amount completed: 99.8% for speech; 99.1% for handstick.
- Data entry speed: 1.336 characters/sec for speech; 0.766 characters/sec for handstick.
- Error rate: 0.033 for speech; 0.09 for handstick.

These results suggest that this speech modality is far better than alternative modalities used in mobile devices that require manual dexterity for alphanumeric data entry. Such modalities are characterized by one or more of the following: decreased physical input area, increased visual scan time, and increased character entry time (e.g., handwriting recognition). A speech user interface similar to *SUITEKeys* appears to be relatively easy to learn and to use, particularly for motor challenged and/or computer illiterate users. Anecdotal evidence from the novice subjects of the study suggests that this system is far less intimidating than other interfaces.

In terms of summative evaluation, we have conducted one small-scale experiment using our latest implementation of *SUITEKeys*. We are also in the process of conducting a large-scale study at the University of Louisiana at Lafayette. The results of the small-scale experiment apply only to speech and are approximate. However they provide an idea as to how closely the current implementation of SUITEKeys is to the ideal Wizard-of-Oz model. These results are as follows:

- Time to completion: 346.8 secs.
- Amount completed: 100%.
- Data entry speed: 0.695 characters/sec.
- Error rate: 0.057.

Discussion

These preliminary summative results suggest that, in its current implementation, *SUITEKeys* is a useful application for motor-impaired users. Although it does not appear to be as effective as handstick (at least in terms of data entry speed and task completion time), it is a viable alternative for users who are unable or unwilling to use means that require manual dexterity for alphanumeric data entry. Of course, this assumes a user with no significant speech impediments and a noise-free environment.

As mentioned above, although the system incorporates a continuous speech engine, this capability could was not

exploited in all cases to improve recognition accuracy. Obviously, discrete speech input slows down data entry.

Research suggests that, in the near future, computer speech-recognition may surpass human speech-recognition [14]. Therefore, the above domain limitations may be specific to the current state-of-the-art in speech recognition. The modular design of SUITEKeys makes it straightforward to improve its speech recognition accuracy by attaching it to newer speech engines, as they become available.

CONCLUSION

SUITEKeys is a continuous-speech user interface for motor-disabled computer users. This interface provides access to all available functionality of a computer by modeling interaction at the level of the physical keyboard and mouse.

By following this "metaphor as model" approach, *SUITEKeys* facilitates the formation of a conceptual model of the system and its linguistic domain [13]. Given that the linguistic primitives of speech user interfaces are invisible (as opposed to the primitives of well-designed graphical user interfaces), this significantly improves the system's usability.

Additional functionality—that is functionality beyond the interface metaphor—is incorporated by making it visible through the *SUITEKeys* graphical user interface. The user may simply read aloud menu entries and dialog-box elements to select them. Further support for learning and retaining how to use the system is provided through context-sensitive help (i.e., "What Can I Say") and progressive disclosure (i.e., "More Help").

This relatively low-level approach has a potential drawback compared to speech user interfaces employing a higher-level approach, such as NaturallySpeaking. In many cases, a user may be required to "synthesize" higher-level actions (e.g., Drag & Drop) through a sequence of low-level actions. This is especially "painful" in the context of dictation. Although *SUITEKeys* does not "claim" to be a dictation package, it address this drawback through (a) its word auto-completion mechanism and (b) its capability to relinquish control to other speech-enabled applications.

Although speech is not the best modality for all human-computer interaction tasks, when delivered at the level of keyboard and mouse it allows for universal access to computing devices—functionally similar to the one enjoyed through a standard QWERTY keyboard and mouse. The presented solution, in a future incarnation, might complement or even replace existing mobile computing modalities in many application domains. Since it does not require much physical device area for alphanumeric data entry (only microphone and perhaps speaker, for feedback), the physical device may shrink as much as advances in microelectronics allow.

ACKNOWLEDGMENTS

This research has been partially supported by the Louisiana Board of Regents grant BoRSF-(1997-00)-RD-A-31. The authors acknowledge the contribution of Rao Adavikolanu, Huong Do, Corey Gaudin, Garrick Hall, William Jacobs, Jonathan Laughery, Michail Lagoudakis, Eric Li, Adi Sakala, and the Spring 1999 Human-Computer Interaction class at the University of Louisiana at Lafayette.

REFERENCES

1. Manaris, B. and Harkreader, A. 1998. SUITEKeys: A speech understanding interface for the motor-control challenged, in *Proceedings of ASSETS '98* (Marina del Rey, CA, April 1998), ACM, 108–115.
2. Manaris, B., MacGyvers, V., and Lagoudakis, M., "Speech Input for Mobile Computing Devices," *Proceedings of 12th International Florida AI Research Symposium (FLAIRS-99)* (Orlando, FL, May 1999), 286-292.
3. Manaris, B. and MacGyvers, V. "Speech for Alphanumeric Data Entry – A Usability Study," submitted to the International Journal of Speech Technology.
4. Goldstein, M., Book, R., Alsio, G., Tessa, S. 1998. Ubiquitous Input for Wearable Computing: QWERTY Keyboard without a Board, in *Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices* (Glasgow, Scotland, 1998), GIST Technical Report 98/1. Available at www.dcs.gla.ac.uk/~johnson/papers/mobile/HCIMD1.html
5. MacKenzie, I.S., Zhang, S.X., Soukoreff, R.W. Text Entry Using Soft Keyboards. *Behaviour & Information Technology 18* (1999), pp. 235-244. Available at www.yorku.ca/faculty/academic/mack/BIT3.html
6. Dragon Systems, Inc., NaturallySpeaking. Available at www.dragonsys.com .
7. Microsoft, MS Speech Engine 4.0. Available at www.microsoft.com/IIT/download/ .
8. Manaris, B. and Harkreader, A. SUITE: speech understanding interface tools and environments, in *Proceedings of FLAIRS '97* (Daytona Beach, FL, May 1997), 247-252.
9. Manaris, B. and Dominick, W.D. NALIGE: A user interface management system for the development of natural language interfaces, *International Journal of Man-Machine Studies 38*, 6 (1993), 891–921.
10. Weiser, M. The world is not a desktop. *Interactions*, January 1994, p. 7.
11. Copestake, A. Applying Natural Language Processing Techniques to Speech Prostheses. *1996 AAAI Fall Symposium on Developing Assistive Technology for*

People with Disabilities. Available at www-csli.stanford.edu/~aac/papers.html .

12. Furnas, G.W., Landauer, T.K., Gomez, L.M, and Dumais, S.T. Statistical semantics: analysis of the potential performance of key-word information systems, *The Bell System Technical Journal 6* (1983), p. 1796.
13. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. *Human-Computer Interaction.* Addison Wesley, Reading, Massachusetts, 1994, p. 146.
14. USC News Service, Machine Demonstrates Superhuman Speech Recognition Abilities, news release 0999025, Sep. 30, 1999. Available at http://uscnews.usc.edu/newsreleases/ .