Speak for Me! Description

Overview

Our users are paralyzed and unable to speak. Due to accident, stroke, or disease, all are confined to either a hospital bed or a wheelchair. Some are blind. The Speak for Me! software enables users to build messages by spelling each word. It presents letters in different orders, guesses words, sentences and commands, and takes action in response to user selections. Users also control the software via "commands" that they spell. They spell these special words and select the commands when the software guesses that they are trying to spell them. These commands address and send messages via email, send them to another user on a local network nursing station, make them appear on a nurse's pager, or display them on the "message board" of the users' own machines. They will also fetch and read email messages, control small applicances for the user, and control the message construction.

Under normal conditions, when a user is not interacting with the program, it is idling, waiting for the users to signal with a single click on a wireless mouse. When they wish to use the program, they click the mouse button. The program assumes that they wish to spell a message or command, and starts displaying or speaking the letters of the alphabet. The general interaction is described in the "Build a Message" conversation below.

Building a Message

How the software responds to a user's selections depends on the current "state" of the word that she is constructing. When idling, a single click will cause the alphabet to be spoken, beginning with the first letter of the alphabet. After she selects a letter, though, the first thing she sees or hears is a "space". Selecting the space ends the current word and begins another. The software does not guess any words or sentences or commands until she has selected at least two letters. Otherwise, there are too many possibilities to make reasonable guesses. There general activities in building a message are described in the "conversation" table below.

Conversation: Build a Message

Actor Actions	System Responsibilities
Click to start speaking	
	Start building a message
Repeat actions until the message is built	
	Determine what to present (from letters, words, sentences, commands, and space)
	Sort the choices (most likely first)
	Present choices to the user
(Optionally, click to select what was presented)	
	Determine action to take based on selection
	If the user has selected a letter, then add the letter to the message
	If the user has selected a word, then strip the partial word and add the selected word
	If the user has selected a sentence, then strip the partial sentence, and add the selected sentence
	If the user has selected the "end sentence" command, then start "Repeat actions" with a new sentence
	If the user has selected another command (e.g. "send message") then quit building the message
	Otherwise, add selection to message
End actions when message is built	
	Process the selected command

Presenting Letters

We normally think of letter order as A .. Z. In “Speak for Me!” this is not always the case. A user can choose from a few different orderings:

- alphabetic (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z)
- frequency (E, T, O, A, N, I, R, S, H, D, L, C, F, U, M, P, Y, W, G, B, V, K, X, J, Q, Z)
- either of the above plus dynamic reordering based on onset coocurrence (see section on Onset Coocurrence)

Guessing Words, Sentences and Commands

As a user spells, the software tries to “guess” words, sentences, and commands as it sees more and more letters. It matches her partial constructions against complete words, sentences and commands in program dictionaries. (Guessing is an area where the application can evolve: the application can use knowledge of word structure and linguistic rules of coocurrence of letters to be “smarter” as it presents letters. It can learn the patterns of the user’s word and sentence constructions to anticipate her choices, and it can use the knowledge of grammar to better predict words within a sentence.)

Commands

A message may contain more than one sentence. So, if after finishing the first sentence, how does a user indicate that she wants to build and add a second sentence? She considers the first sentence complete, but doesn’t want to send it yet because she has more to say! As in all other operations that she controls, she builds a command word. The command she uses to terminate a sentence and add it to the message under construction is “ES” (for “end sentence”). So, she begins building a new word, this time starting it with “SE”. Each time the user makes selects a letter, the software attempts to “guess” what she is trying to say. In this case, it knows that there is a command word that begins with “SE”, so it presents the phrase “End the sentence” to her. If she selects it upon seeing or hearing it, the software ends the sentence and begins another. If she doesn’t select “end the sentence”, it continues building her word that begins with “SE”.

Anytime that a user selects a command, the software will execute the command. The command will often use the current message as it performs its purpose, e.g., send the message, display the message, etc. This command word approach is very flexible and lets us avoid an “in-your-face” style of interaction with lots of spoken menus and submenus.

Available Commands

The commands enable a user to perform actions with the software. If she wishes to read her email inbox messages rather than construct a new sentence, she spells the command word for read email (RE), the selects the command when she hears it. The flexibility of the command word approach will allow her to

- end the current sentence and continue to the next (spell ES and select “end the sentence”)
- end the message and start addressing it (spell AM and select “address the message”)
- read (or listen to) email (spell RE and select “read email”)
- clear the message board and idle (spell CB and select “clear message board”)
- review the message board (spell RM and select “review message board”)
- call for help in emergency! And idle (spell EE and select “call for help”)

The software will guess certain commands when it thinks them relevant. For example,

- send the message (after each address selection)

Onset Coocurrence

There are special rules that apply to word construction at the beginning of a word. In what is called the onset of the first syllable, there are only a few letters that may follow the first letter. This onset coocurrence provides us with a relatively simple way to predict the second and third letters of a word, given the first, or first and second letters, respectively. This enables us to present the most probable letters at the beginning of the series, followed by the rest of the alphabet in the ordering of the user's choice. Here is a table of the recognized onset coocurrences:

Word begins with …	Likely next letters …
B	R or L
C	H, L, or R
D	R or W
F	R or L
G	R or L
K	N
P	R, L, H, or S
Q	U
R	H
T	H, R, or W
W	H or R
ST	R
SC	H or R
SH	R
SQ	U
PH	R
TH	R
CH	R or L

For example, when a user is trying to spell the word "bring", onset coocurrence rules indicate that the initial b will often be followed by the consonants l or r. After she chooses the initial "B", we present the two consonants "R" and "L" first, followed by the rest of the alphabet. Further, if we can make the construction of the first few letters efficient (as we do when we use onset coocurrence), we quickly increase the probability that we will be able to guess the correct word from the dictionary. The number of matches will decrease rapidly as we build the word past the first two or three letters.

Suspending

If a user is interrupted during her construction of a message, she should be able to divert her attention to her nurse or visitor without fearing that she will lose her current work. If, at the end of the presentation of the alphabet, she has not chosen a letter, the software suspends. When she wants to continue building her message, she starts the software as she usually does, with a single click. Since it has been "suspended" by reaching the end of the alphabet without her making a selection, the program reads the full message to her, up to but not including the last word. It then spells the last, partial word to her, announces that she is resuming at that point, and starts normal operation for completing the latest word.

Sending a Message

When the message is complete, she will send it to a destination. All destinations (email addresses, etc.) have aliases (typically, the names of people, groups of people, or places) so that she can easily understand the choices and designate where the message should be delivered. The availability of certain destinations depends on the services that are available at her installation. At a minimum, all installations will be able to send messages to her "Message Board", which is the alias for an area of the display screen on her computer, and all installations will offer email. So there will always be a list of user-defined names that are aliases for email addresses or mailing lists of friends and family. If the nursing home or hospital has a computer network, there may be a "Nursing Station", the alias for the network computer at the main desk, as well as a list of names of networked nurses and doctors, each wearing a pager with email capabilities. Once the name of a person, a place, or a mailing list is selected, the program will figure out how to deliver the message and attempt to do so. The message may be delivered to its destinations via email, local area networking, or may be displayed on the local machine.

Configuring the User Interface

Our users vary in their capabilities. Since some can see the screen, the software must be able to display the letters, words, sentences and commands, as well as keep the user informed of the current state of the message that is under construction. For those that are blind, it must be able to speak these same elements to the users.

The software will ship with a highly configurable user interface, the particular configuration depending on the distance from the user that the screen will be located. For example, since some users will have a laptop close by, the screen should be able to display on a single screen a rich set of information in a relatively small font visible from 2 feet or so. For those users that must locate their machines across the room on a table, the screen will be able to display only a small amount of information at one time, in a font visible from 8 or 10 feet. All users can command the software to turn the speaker on or off and to turn the volume up or down.

Different potential choices that are presented to the user must have different "delays" before presenting the next choice. For example, when a word is guessed, there should be a slightly longer delay before continuing on to present the next choice. In general, anything that might surprise the user should offer a bit longer time for the users to select them. These timing characteristics and any other user preferences should be configurable for the user by a speech therapist, using a preferences screen that runs within the Speak for Me! Software. When displaying words, commands, and sentences, it is sufficient to allow the same amount of time of each type. But when speaking these choices, the software must wait until the choice has been completely spoken, and measure a small delay after it is complete before continuing. For example, displaying the words "ant" and "antidisestablishmentarianism" can take the same amount of time when displayed as text on the screen, but the wait will be much longer when speaking the longer word.

The command-driven approach allows the user to specify directly what they wish to do with the software. But the software should do as much as possible to keep the user informed of their current context without constantly getting in their way of the task at hand. For example, it should beep to inform the user that their selection was recognized and handled, it should voice or display information regarding the success or failure of message delivery, and it should echo that it is "ready and waiting" when it is idling and waiting for the users to begin message construction.