December 15, 2016

Inline voice commands in word processor

Georges Rotival

Victoria Fritz

Kelly Moran

Shawn Lauriat

Kirti Agarwal

Inline voice commands in word processor

ABSTRACT

Invoking a command in a word processor typically involves a break in the flow of writing. For example, if a user wants to insert a table into a document that she is editing, she has to execute an operation such as pressing a key combination, performing a mouse movement, or a gesture to invoke a command. Such breaks in flow causes a loss of productivity. Further, performing such movements may be difficult for users with motor disabilities. This disclosure describes techniques that enable a user to issue voice commands without a break in the flow of writing.

KEYWORDS:

- Voice commands
- Word processor
- Speech-to-text
- Inline command

BACKGROUND

During the editing of a document in a word processor, there is frequent occasion to invoke a command. For example, a user may want to invoke a command to change a font, to change paragraph alignment, etc. Invoking a command typically involves a break in the flow of writing. For example, if a user wants to insert a table into a document that she is editing, she has to execute an operation such as pressing a key combination, performing a mouse movement, or a gesture to invoke a command. Further, such operations may also include a search within a menu to locate the right command. This break in the flow of writing causes a loss in productivity. Some word processors permit a user to toggle between dictation and voice

command modes, which also requires a break in flow to issue commands. The requirement for switching input modes can be difficult for people with motor impairments.

DESCRIPTION

This disclosure describes techniques that enable a user to invoke a command using voice. The commands can be issued inline with dictating text such that there are no separate modes for entering text and voice commands. Per techniques of this disclosure, a command is automatically detected and distinguished from text. An utterance by a user is considered as a possible command, e.g., if it is preceded and succeeded by a pause of at least a certain duration. Such an utterance is matched against a table of commands, and is deemed to be a command if a match of sufficient confidence is found. Thus, in order to issue a command, the user does not need to deviate from the normal speech-to-text or keyboard flow. Rather, the user simply punctuates a spoken phrase with pauses in order to indicate the issuance of a command.

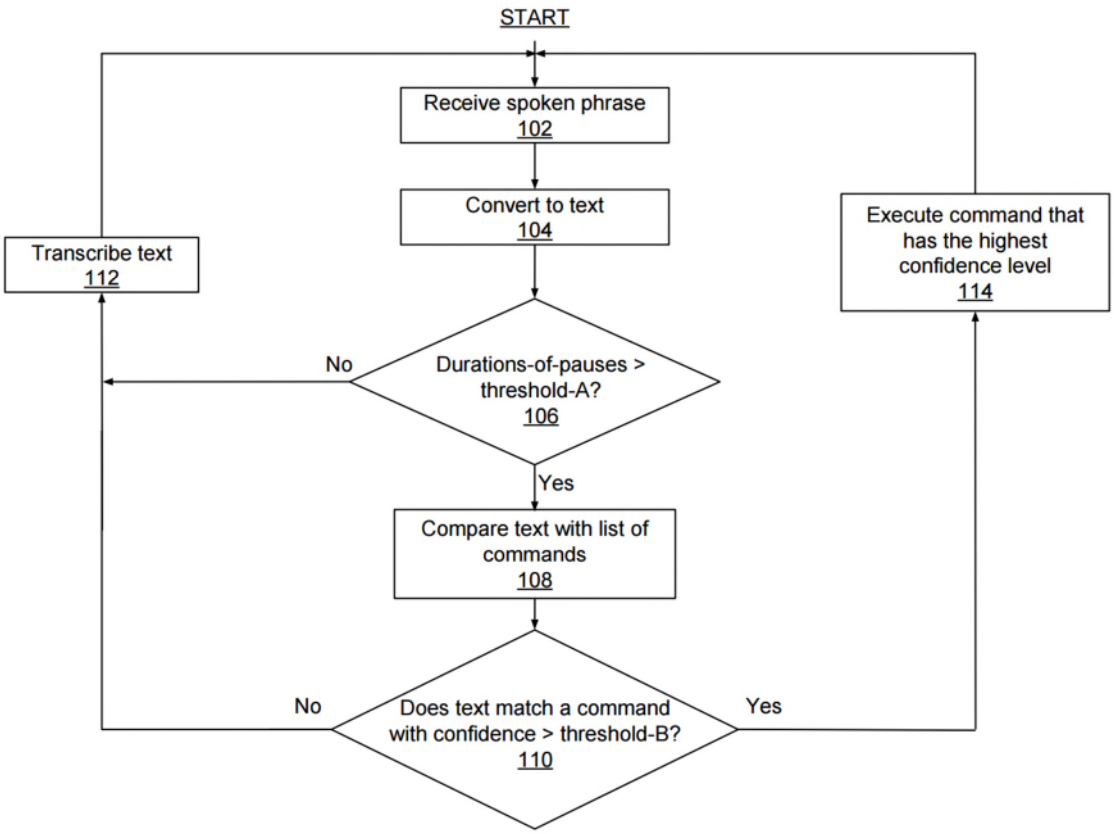

Fig. 1: Differentiating a command from text

Fig. 1 illustrates the steps involved in distinguishing text from command. A phrase spoken by a user is received (102) and converted to text (104). The conversion from speech to text may be done by a speech-to-text engine. The duration of pauses within or around the phrase is examined in step 106 to determine if any two or more pauses exceed a certain threshold, "threshold-A". For example, threshold-A may be one second. If no two pauses exceed threshold-A, then the spoken phrase is deemed to not be a command, and is transcribed to text (112). The system subsequently loops back to step 102, where the next spoken phrase is awaited. If at step 106 at least two pauses exceed threshold-A, then the intervening text may possibly be a command. To determine if intervening text(s) are commands, the intervening text(s) are compared with a table of commands. The comparison of intervening text with the table of commands may be performed by an action matcher, which returns a list of confidence

measurements reflecting the level of match of an intervening text with each command in the table of commands.

If confidence levels corresponding to all commands in the table are below a certain threshold, "threshold-B", (110) then it is deemed that the spoken phrase is not a command, and the text corresponding to the spoken phrase is transcribed as text for insertion into the document (112). If the confidence level corresponding to a command is higher than threshold-B, (110) then the spoken phrase is deemed to include commands. The command with highest confidence level is chosen and executed (114). Execution of a command may include displaying the executed command in an information bubble or window within the word processor.

Examples of Use

The following examples illustrate how a user might task a word processor that is enabled, per techniques of this disclosure, to distinguish between commands and text. As the examples below illustrate, a voice command may include one or more arguments.

Example 1: User wishes to set background color to a certain color, say red. During the dictation of text into the word processor, the user pauses for a second, says "Set background color red", then pauses again for a second, then continues dictating text.

Example 2: User wishes to insert a table of dimension 10×10. During the dictation of text into the word processor, the user pauses, says "Insert table with dimension 10 by 10", then pauses again, then continues dictating text.

Example 3: User wishes to change font to <*fontname*>. During the dictation of text into the word processor, the user pauses for a second, says "Change font to <*fontname*>", then pauses again for a second, then continues dictating text.

Certain commands, e.g., the "undo" command, the "redo" command, commands applied to items that are selected, etc. require storage of the state of the document. This is illustrated by the examples below.

Example 4: User wishes to boldface the last word ("dolor") of existing text "Lorem ipsum dolor". Therefore she selects "dolor" and then says "boldface". Initially, the word processor overwrites the selected word "dolor" with transcribed text "boldface" to update the text to "Lorem ipsum boldface". At this point, the word processor determines that the last spoken word "boldface" was a command. The replacement of "dolor" by "boldface" is first undone, and the boldface command is applied to "dolor" to produce the correct transcription "Lorem ipsum **dolor**". In this case, the word processor stores and subsequently recovers the state of the document to correctly perform the operation specified by the user.

Example 5: User has thus far entered "Lorem ipsum **dolor**". The last action thus far of the user was to format the word "dolor" in boldface. User now wishes to undo the last action. The user therefore says "undo". Initially, the word processor transcribes the user's speech as "Lorem ipsum **dolor** undo", and then determines that the last utterance "undo" specifies a command. At this point, the word processor first deletes the last word transcribed (namely "undo") and then applies the undo command to "**dolor**". In this manner, the word processor produces the correct transcription "Lorem ipsum dolor" with the boldface operation on the word "dolor" reversed in accordance with the undo command. In this case, the word processor stores and recovers the state of the document to correctly perform the operation specified by the user.

Example 6: User has thus far entered "Lorem ipsum dolor", changed the font of "dolor" to boldface, and then changed it back to normal font. The user now wishes to redo the boldface

operation on “dolor”. The user therefore says “redo”. The word processor transcribes the user’s speech as “Lorem ipsum dolor redo”, and then determines that the last utterance “redo” was a command. The word processor proceeds to delete the last word transcribed (namely “redo”), and then applies the redo command to the word “dolor” to change it to “**dolor**”. By saving and restoring document state, the redo command is executed to obtain the result desired by the user.

In implementing the word processor, data structures are used to store, recover, and reset states of a document that is being edited, e.g., in a stack. Such data structures are edited by program code that correctly applies the commands spoken by a user. Certain rules are followed to correctly perform undo, redo, and other operations that require storage and retrieval of documents states, such as:

1. The saved state index is obtained from the bottom of the undo stack. This is because new edits are pushed onto the top of the undo stack, thereby changing the index of the top of the stack. It is only the index of the bottom of the undo stack that is constant.
2. The saved state index is updated only if the undo stack is spliced for indices between the bottom of the undo stack and the saved state index.
3. While a state is stored, the redo stack is not cleared, even if a new edit result is pushed.
4. If a new edit is pushed into the undo stack after a save operation and a reset is triggered, then the redo stack is cleared.
5. If a save operation is followed by a reset without any undo-able edit in between, then the redo stack is not cleared.
6. An undo/redo operation that takes place while a state is saved triggers a reset. This is because the state of the undo/redo stack in such a situation may be ambiguous and difficult to track.

Techniques of this disclosure, when implemented in an online word processor e.g., provided via an internet browser, enable the user to take advantage of the latest suite of voice commands, or features registered as voice commands. New commands (e.g., corresponding to addition of features in the word processor) or updated commands (e.g., when an interaction redesign, such as a menu update, is carried out) can easily be added as voice commands, without substantial code updates. This is because the provider of the word processor can ensure that the table of voice commands is always up-to-date, so that the user does not need to wait for a new release of software in order to get the latest commands.

CONCLUSION

Techniques of this disclosure enable a word processor to accept voice commands inline with text, without the need to switch from text input (via keyboard, dictation, etc.) to command input (via keyboard, mouse, voice command, etc.) A user need not break the flow of writing, or speaking, into the word processor in order to issue a command. A command is distinguished from text by determining the durations of pauses around spoken phrases, and by comparing phrases that may possibly be commands with a table of commands. If a spoken phrase is found to include a command, then the command is executed, and user informed of the execution using an information bubble. If a spoken phrase is found not to include a command, it is transcribed as text within the word processor. Techniques of this disclosure are useful not only to improve productivity, but also serve as an accessibility tool, e.g., for users with motor impairments.