Proceedings of HCI International 2001, Vol.1, pp.198-202, August 2001.

Speech Completion: New Speech Interface with On-demand Completion Assistance

Masataka Goto, Katunobu Itou, Tomoyosi Akiba, and Satoru Hayamizu

National Institute of Advanced Industrial Science and Technology (former Electrotechnical Laboratory)
1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, JAPAN.
m.goto@aist.go.jp

Abstract

This paper describes a novel speech interface function, called *speech completion*, that helps a user enter a word or phrase by *completing* (filling in the rest of) a phrase fragment uttered by the user. Although the concept of completion has been widely used in text-based interfaces, effective completion for speech has not been proposed. We enable a user to invoke the speech-completion function intentionally and effortlessly by building an interface that displays completion candidates when a filled pause is uttered (a vowel is lengthened) during a phrase. The filled pause can be considered a nonverbal modality that has not been used in speech input interfaces. In our experience with a system that includes a filled-pause detector and a speech recognizer capable of listing completion candidates, the effectiveness of speech completion was confirmed.

1 INTRODUCTION

Current speech-input interfaces have not fully exploited the potential of speech. Although human speech has two aspects, verbal information (e.g., words) and nonverbal information (e.g., hesitation), most speech recognizers utilize only the modality of verbal information. They are therefore, as it were, nothing more than a computer keyboard that sometimes makes key-recognition errors; even if the precision of speech recognizers could be improved, it is difficult to build an interface that is handier than a keyboard. The purpose of this study is to build a speech interface that makes full use of the role nonverbal speech information plays in human-human communication.

From among various nonverbal information, we focus on a filled pause (the lengthening of a vowel), which is a hesitation phenomenon and is apt to reflect the mental state of a speaker (e.g., trying to think of a subsequent word) (Takubo, 1995; Rose, 1998). When a speaker cannot remember an entire phrase and hesitates with a filled pause, the listener sometimes helps the speaker recall it: the listener suggests options obtained by *completing* the partially uttered fragment (i.e., by filling in the rest of it). For example, when a speaker cannot remember the Japanese phrase *"maikeru jakuson"* (in English, "Michael Jackson")[1] and stumbles, saying *"maikeru–"* ("Michael–") with a filled pause *"ru–"* ("l–"), a listener can help the speaker by asking whether the speaker intended to say *"maikeru jakuson"* ("Michael Jackson"). Although one of the reasons that speech communication is comfortable is that we can expect a listener to help us this way when we utter vague or incomplete information, this phenomenon has been given little attention in speech recognition research.[2]

The concept of *completing* a fragment has been widely used in text-based interfaces. For example, several text editors (e.g., Emacs) and UNIX shells (e.g., tcsh and bash) provide functions completing the names of files and commands. These functions fill in the rest of a partially typed fragment when a *completion-trigger key* (typically the Tab key) is pressed. Completion functions for pen-based interfaces, such as POBox (Masui, 1998), have also been proposed. Even though completion is so convenient that it becomes indispensable to those who have used it, effective completion functions for speech input interfaces have not been developed because there has been no way to trigger them during natural speech input.

In this paper we describe a completion function, called *speech completion*, that enables a user to enter a word or phrase by uttering a fragment of it with a filled pause. The following sections explain the basic concept of speech completion, describe the design and implementation of a speech recognition interface with the speech-completion function, show experimental results obtained with the interface, and discuss directions of future work.

[1] When a foreign name like "Michael Jackson" is written or pronounced in Japanese, the Japanese style is used: *"maikeru jakuson."*

[2] In most current speech recognizers, it is understood that the user prepares the content in advance and pronounces it carefully and precisely. Although hesitation phenomena such as filled pauses and restarts occur frequently in spontaneous speech, current recognizers accept only fluent speech without these phenomena because they tend to cause recognition errors. In addition, only few attempts have been made at making good use of the valuable roles these phenomena play in human-human communication.

2 SPEECH COMPLETION

Speech completion, the general term for interface functions that enable a user to invoke completion assistance during speech input, has the following three advantages:

1. It helps the user recall uncertain phrases.
2. It saves labor when the input word or phrase is a long one.
3. It reduces the psychological pressure of being forced to utter the whole content carefully and precisely, something that most current speech recognizers force users to do.

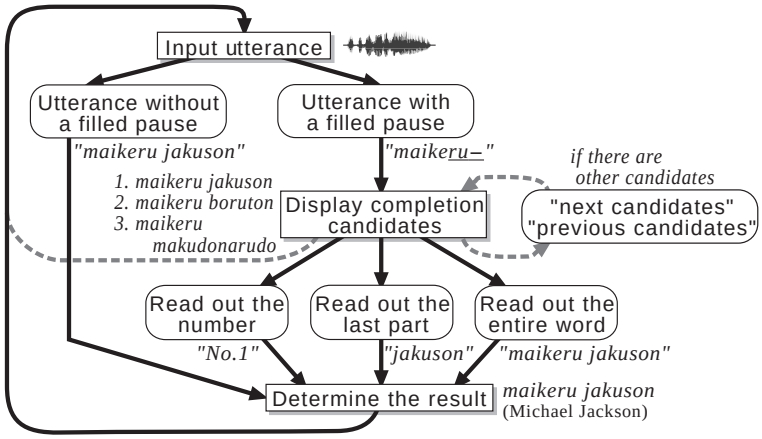


Figure 1: Flowchart for our speech input interface with speech completion assistance.

Since completion should not become annoying, it should be invoked only when the user wants to obtain completion candidates. For effective and practical speech completion, we therefore use an intentional filled pause to trigger a speech-completion function that fills in the rest of a partially uttered fragment. Since the filled pause is a very natural trigger, the user can invoke the speech-completion function intentionally and effortlessly. This is especially true for Japanese, a moraic language in which every mora ends with a vowel that can be lengthened. In fact, speakers typically use filled pauses to gain time to recall a word or to wait for a listener to help with word choice.

Although various completion levels — such as those of the word, phrase, clause, and sentence — can be considered, in this paper we concentrate on word-level and phrase-level completion (this completion can be naturally extended to the sentence level). We deal with words registered in the system vocabulary of a speech recognizer, and phrases such as the names of musicians and songs can be registered as single words.

3 SPEECH INPUT INTERFACE WITH SPEECH COMPLETION ASSISTANCE

We designed a speech-input-interface system (Figure 1) that can assist a user by completing any system vocabulary word (which can be either a single word or a phrase). It works as follows:

1. When the user does not remember the last part of a word or phrase and utters the first part while intentionally lengthening its last syllable (making a filled pause),[3] the system displays a numbered list of completion candidates whose beginnings acoustically resemble the uttered fragment.

   When, for example, the Japanese phrase *"maikeru jakuson"* (in English, "Michael Jackson") is registered as one word in the system vocabulary, a user uttering the fragment *"maikeru–"* ("Michael–") gets completion candidates such as *"maikeru jakuson"* ("Michael Jackson"), *"maikeru boruton"* ("Michael Bolton"), and *"maikeru makudonarudo"* ("Michael McDonald").
2. The user can invoke the display of other candidates by uttering the turning-the-page phrases, "next candidates" and "previous candidates," displayed whenever there are too many candidates to fit on the computer screen. If all the candidates are inappropriate or the user wants to enter another word, the user can simply ignore the displayed candidates and proceed with the next utterance.
3. When the user selects one of the candidates by uttering (reading out) either its number, the last part of the word, or the entire word, that word is highlighted and used as the speech input result.

4 IMPLEMENTATION

Figure 2 shows the architecture of our speech-completion system. The boxes in the figure represent different processes, and the four main processes are those of the filled-pause detector (Section 4.1), the speech recognizer (Section 4.2), the interface manager, and the graphics manager. Those processes can be distributed over a LAN (Ethernet) and connected by using a network protocol called *RVCP (Remote Voice Control Protocol)*, which is an extension of RMCP (Goto, Neyama, & Muraoka, 1997). RVCP supports timestamp-based synchronization for real-time information handling and supports efficient multicast-based information sharing without the overhead of multiple transmission.

[3]The user can insert a filled pause at an arbitrary position while uttering a word or phrase.

The filled-pause detector controls the two modes of the speech recognizer, the *normal mode* and the *completion mode*. In the completion mode triggered by a filled pause, the recognizer generates a numbered list of completion candidates that is sent to the interface manager managing the state transition of the interface (flowchart shown in Figure 1). The graphics manager manages a front-end GUI and displays on the screen the recognition results and a pop-up window containing the candidate list.

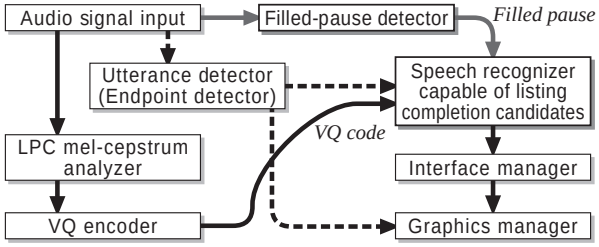


Figure 2: System architecture.

4.1 Filled-pause detector

Because speech completion is impractical without a real-time method for detecting filled pauses that is independent of vocabulary and language, we use a robust filled-pause detection method (Goto, Itou, & Hayamizu, 1999). It is a bottom-up method that can detect an intentionally lengthened vowel in any word without using top-down information (language model). It determines the beginning and end of each filled pause by finding two acoustical features of filled pauses, small fundamental frequency transitions and small spectral envelope deformations. Those features are evaluated by using a sophisticated instantaneous-frequency-based analysis (Goto et al., 1999). Figure 3 shows an example of a detected filled pause.

Note that, as shown in Figure 2, the processing of the real-time filled-pause detector that directly analyzes the input audio signal is executed in parallel with that of the following HMM-based speech recognizer.

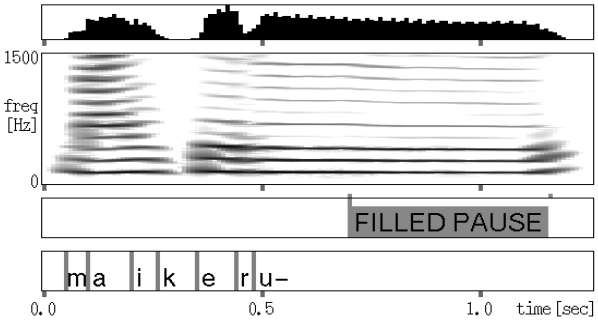


Figure 3: An example of a detected filled pause for "maikeru–"*: the power and the spectrum (top), the filled pause (middle), and the phoneme sequence recognized by the speech recognizer that inhibits the transition from a vowel phoneme to the next phoneme during the detected filled pause (bottom).*

4.2 Speech recognizer capable of listing completion candidates

To provide a list of completion candidates whenever a filled pause was detected, we extended an HMM-based speech recognizer, *niNja* (Itou, Hayamizu, & Tanaka, 1992). The extended recognizer receives VQ codes and the detected filled pause and sends the following results to the interface manager:

- The top N_{choice} completion candidates
  The completion candidates are generated at the beginning of each filled pause and are sent immediately (i.e., before the end of the current utterance). Each candidate includes the information of which parts (phonemes) of it have been uttered (how much of it has been uttered).
- The top N_{result} recognition results
  At the endpoint of each utterance, the recognition results of that utterance are sent.

The recognizer uses not only a vocabulary of words to be input and be completed (e.g., the names of musicians and songs) but also a vocabulary for operating the interface (e.g., the candidate numbers and the turning-the-page phrases). All these words are stored in a tree structure as shown in Figure 4, where the wedge marks represent multiple hypotheses maintained by a frame-synchronous Viterbi beam search decoder.[4]

The recognizer enters the completion mode when the beginning of a filled pause is detected (about 200 ms after a vowel is lengthened). It generates candidates by tracing from the top N_{seed} hypotheses (at the beginning of the pause) to the leaves (Figure 4): the candidates are obtained by deriving from the vocabulary tree those words that share the prefix corresponding to each incomplete word hypothesis of the uttered fragment. The top N_{choice} candidates (leaves) are sorted and numbered in order of likelihood and then sent to the interface manager. We call the nodes corresponding to the top N_{seed} hypotheses the *speech completion seeds*. For example, if the top black circle in Figure 4 is a seed, the completion candidates obtained are "Michael Bolton" and "Michael Jackson."

[4]Because single phonemes cannot be recognized accurately enough, most up-to-date speech recognizers do not determine the phoneme sequence of a word phoneme by phoneme but instead choose the maximum likelihood hypothesis while pursuing multiple hypotheses predicting the next phoneme.

To enable the user to select the correct candidate by reading out the last part of it, the speech-completion system must be able to recognize last-part fragments that are not registered as vocabulary words. We therefore use an *entry node table* in which are listed the roots (nodes) from which the decoder starts searching. In the normal mode, only the root of the vocabulary tree is listed. During the utterance just after the listing of completion candidates, speech completion seeds are temporarily added to the table as shown in Figure 4. Although a candidate can be selected by uttering just the last part, the recognition result sent to the interface manager is the entire word.

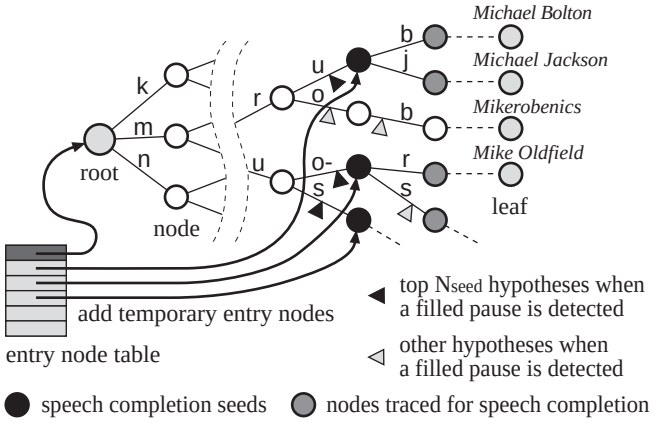


Figure 4: Obtaining completion candidates on the vocabulary tree at the beginning of a filled pause and adding speech completion seeds to the entry node table.

5 EXPERIMENTAL RESULTS

We tested the system with a system vocabulary comprising about 500 names of musicians and songs in Japanese. Our current implementation uses the following parameter values, which should be adjusted according to the vocabulary and the purpose of speech input interface: N_{choice} = 20, N_{result} = 5, and N_{seed} = 5. Figure 5 shows an example of the graphics output displayed during speech completion.

Experimental results showed that our system can provide a helpful list of completed full names when a filled pause is uttered. We have confirmed that the speech-completion function is intuitive enough to be used without any training and is effective for entering uncertain phrases. The completion candidates are especially useful when the input phrase is a long proper name.

6 DISCUSSION

Our speech completion research began with the intention of making speech-recognition technologies user-friendly. This approach suggests various directions for future work.

6.1 Autocompletion functions

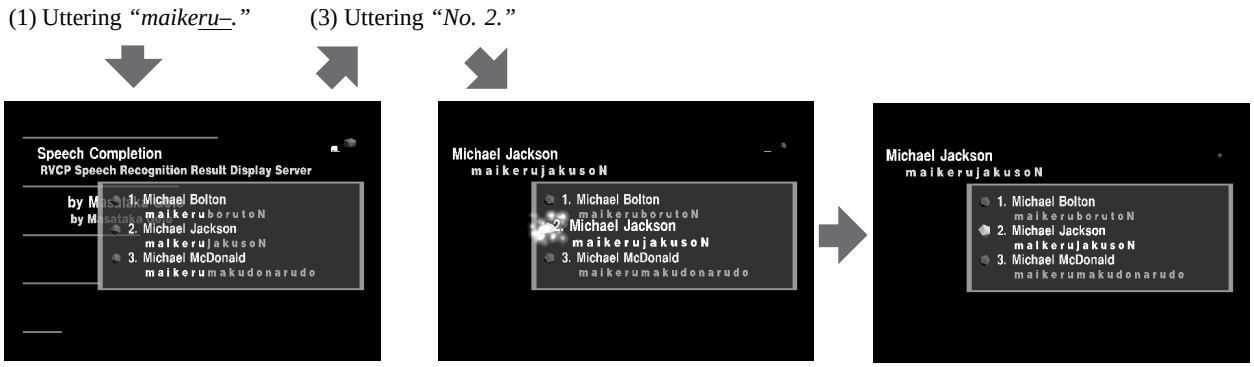
While text-based (keyboard-based) completion functions manually invoked by a completion-trigger key were mentioned in Section 1, autocompletion functions have also been used in Reactive Keyboard (Darragh, Witten, & James, 1990) and for URL entry on web browsers. These autocompletion functions automatically list up completion candidates every time a user types a key. The interface with such functions is also called *predictive interface* and is considered effective.

Such autocompletion functions, however, are not suitable for use in speech input interfaces. In keyboard input, there is no ambiguity in recognizing which key is pressed and the boundaries between successive characters are obvious. In speech input, on the other hand, the recognition of each phoneme is ambiguous and the boundaries between successive phonemes are not easily determined.[5] It is therefore hard to determine when autocompletion candidates should be displayed. Even if the candidates are displayed at arbitrary regular intervals, it is very difficult to continually provide candidates as good as those obtained in text-based autocompletion. The autocompletion of speech input is thus likely to become annoying and not be practical since a predictive interface tends not to be used when it is less convenient than an interface that is not predictive. In our speech-completion function, by enabling a user to explicitly invoke the completion function by using an intentional filled pause, we have successfully built a practical interface that does not interfere with the user at all when the completion is not needed.

6.2 Multimodal interface using multiple modalities in speech audio signals

As stated in Section 1, most current speech interfaces use only the modality of verbal information that can be handled by speech recognition. Our speech-completion function, on the other hand, achieves a user-friendly interface because it exploits the modality of nonverbal information — the filled pauses. We regard our speech completion as a multimodal interface that makes use of multiple modalities contained in speech audio signals.

[5]Metaphorically speaking, speech is not block letters but cursive letters that are not easily segmented.



(2) A pop-up window containing completion candidates appears.

(4) The second candidate is highlighted and bounces.

(5) The selected candidate *"maikeru jakuson"* is determined to be the recognition result.

Figure 5: Screen snapshots of speech completion when the phrase "maikeru jakuson" *("Michael Jackson") is entered.*

Starting from our speech-completion interface, interfaces that are more user-friendly could be built by introducing other nonverbal modalities. In contrast to a computer keyboard, the current speech recognizers have dealt with only a part of the normal letter keys.[6] In this study, on the other hand, the role of the special key "Tab" (the typical completion-trigger key in text-based interfaces such as UNIX shells and the Emacs editor) is triggered by the filled pause. This approach opens up new vistas for future research that assigns other nonverbal information (e.g., pitch and speech rate) to special keys.

Furthermore, speech interfaces can go beyond the limitations of computer keyboard functions because speech has both verbal and nonverbal modalities that can naturally and simultaneously provide different functions. In our speech-completion function, for example, the voice during a filled pause simultaneously conveys both phoneme information (verbal modality) and the user's mental state (nonverbal modality); this function can be considered more efficient and natural than text completion using the completion-trigger key. Typical nonverbal modalities can thus provide meta communicative functions that make ordinary verbal communication rich. By fully bringing out this kind of potential capabilities of speech, we will be able to build excellent interfaces that enhance well-known advantages of speech interfaces, such as hands-free and fast input speed.

7 CONCLUSION

We have described the new speech interface function *"speech completion,"* which fills in the missing part of a partially uttered fragment in order to help a user enter an uncertain phrase. While we have confirmed the effectiveness of this function for the task of inputting the names of musicians and songs, it can also be immediately applied to various other speech applications. We believe it will become as indispensable in speech interfaces as text completion is in good text-based interfaces.

REFERENCES

Darragh, J. J., Witten, I. H., & James, M. L. (1990). The Reactive Keyboard: A Predictive Typing Aid. *IEEE Computer*, *23*(11), 41–49.

Goto, M., Itou, K., & Hayamizu, S. (1999). A Real-time Filled Pause Detection System for Spontaneous Speech Recognition. In *Proceedings of Eurospeech '99*, pp. 227–230.

Goto, M., Neyama, R., & Muraoka, Y. (1997). RMCP: Remote Music Control Protocol — Design and Applications —. In *Proceedings of International Computer Music Conference*, pp. 446–449.

Itou, K., Hayamizu, S., & Tanaka, H. (1992). Continuous speech recognition by context-dependent phonetic HMM and an efficient algorithm for finding N-best sentence hypotheses. In *Proceedings of ICASSP 92*, pp. I–21–24.

Masui, T. (1998). An Efficient Text Input Method for Pen-based Computers. In *Proceedings of CHI'98*, pp. 328–335.

Rose, R. L. (1998). The communicative value of filled pauses in spontaneous speech. Master's thesis, University of Birmingham.

Takubo, Y. (1995). Towards a Linguistic Model of Speech Performance (*in Japanese*). *Journal of Information Processing Society of Japan*, *36*(11), 1020–1026.

[6] In this paper, we consider that a computer keyboard consists of two parts, the normal letter keys for typing alphabetical and numeric characters and the other special keys such as Tab, Delete, and Escape keys.