Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP-2002), pp.1489-1492, September 2002.

SPEECH COMPLETION: ON-DEMAND COMPLETION ASSISTANCE USING FILLED PAUSES FOR SPEECH INPUT INTERFACES

Masataka Goto, Katunobu Itou, and Satoru Hayamizu

National Institute of Advanced Industrial Science and Technology (AIST)
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 is widely used in text-based interfaces, there have been no reports of completion being effectively applied to speech. By using a filled pause, we enable a user to effortlessly invoke the speech-completion function which helps the user recall uncertain phrases and saves labor when the input phrase is long. When a user hesitates by lengthening a vowel (a filled pause is uttered) during a phrase, our system immediately displays completion candidates whose beginnings acoustically resemble the uttered fragment so that the user can select the correct one. In our experiments with a system that included a filled-pause detector and a speech recognizer capable of listing candidates, the effectiveness of speech completion was confirmed.

1. INTRODUCTION

One reason why human-human speech communication is comfortable is that we can expect a listener to help us when we utter vague or incomplete information. In Japanese, when a speaker cannot remember an entire phrase and hesitates while uttering it, a listener sometimes helps the speaker recall it: the listener suggests options by *completing* the partially uttered fragment (i.e., by filling in the rest of it). For example, when a speaker cannot remember the last part of a Japanese phrase *"maikeru jakuson"* (in English, "Michael Jackson")[1] and stumbles, saying *"maikeru–"* (in English, "Michael, uh..." or "Michael–") with a filled pause *"ru–"* ("uh..." or "–"),[2] a listener can help the speaker by asking whether the speaker intends to say *"maikeru jakuson"* ("Michael Jackson"). The purpose of this study is to improve the usability of speech input interfaces by providing this *completion* assistance.

The concept of completing a fragment is 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 [1], have also been proposed. However, even though completion is so convenient that it often becomes indispensable to users, effective completion functions for speech input interfaces have not been developed because there has been no way to trigger them during natural speech input.

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

[2] In Japanese, vowel-lengthening hesitations like "maikeru–" (sounding like "Michael–" in English) are very common, while inserting-filler hesitations like "Michael, uh..." are usually used in English.

In this paper we describe a completion function for speech input, called *speech completion*, that fills in the rest of a partially uttered fragment of a word or phrase. The most important point is that we use an intentional filled pause (the lengthening of a vowel during hesitation) to trigger this speech-completion function. To prevent this kind of completion assistance becoming annoying, it should be invoked only when a user wants to obtain completion candidates. Because the filled pause is a natural hesitation that indicates a user is having trouble thinking of or recalling a subsequent word or phrase, its use makes this speech-completion function effective and practical. The speech-completion function can also be considered a new way to exploit nonverbal speech information (i.e., a filled pause); the filled pause has not been positively used in speech recognition although it plays valuable roles in human-human communication.

In the following sections, we explain the basic concept of speech completion and then describe the design and implementation of a speech recognition interface with the speech-completion function. Finally, we show that experimental results from forty-five subjects indicated the effectiveness of speech completion.

2. SPEECH COMPLETION

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

1. A user can more easily recall uncertain phrases.
2. Less labor is needed to input a long word or phrase.
3. The user is not forced to utter the entire content carefully and precisely, as is required by most current speech-recognition systems.

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

Since a user may want to complete speech in either direction (forward or backward), we propose two completion methods:

1. *Forward speech completion*
   A user who does not remember the last part of a word or phrase can invoke this completion by uttering the first part while intentionally lengthening its last syllable (making a filled pause). The user then gets completion candidates obtained by filling in the last part.
   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–" or "Michael, uh...") gets completion candidates

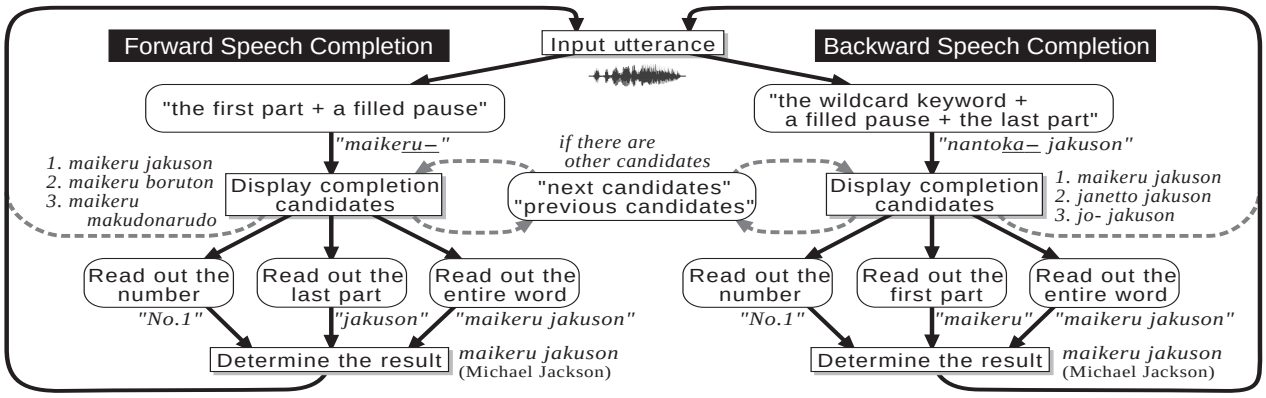

Fig. 1. *Flowchart of the speech input interface with forward and backward speech completion.*

such as *"maikeru jakuson"* ("Michael Jackson"), *"maikeru boruton"* ("Michael Bolton"), and *"maikeru makudonarudo"* ("Michael McDonald").

2. *Backward speech completion (Wildcard speech completion)*
   A user who does not remember the first part of a word or phrase can invoke this completion by uttering the last part after intentionally lengthening the last syllable of a predefined special keyword — called *wildcard keyword*. Completion candidates are generated by replacing the wildcard keyword (filling in the first part) as if a wildcard search was done.
   When, for example, the Japanese wildcard *"nantoka"* (in English, "something") is defined as the wildcard keyword, a user uttering *"nantoka– jakuson"* ("something– Jackson")[3] gets completion candidates such as *"maikeru jakuson"* ("Michael Jackson"), *"janetto jakuson"* ("Janet Jackson"), and *"jo- jakuson"* ("Joe Jackson").

Note that these two completion methods are triggered by the filled pause that is a typical hesitation phenomenon and is apt to reflect the mental and thinking states of a speaker, such as those in which the speaker is trying to think of a subsequent word [2]. Since the filled pause is a natural trigger,[4] the user can invoke the speech-completion function effortlessly.

3. SPEECH INPUT INTERFACE WITH SPEECH COMPLETION ASSISTANCE

We designed a hands-free speech-input-interface system with forward and backward speech-completion functions. It can assist a user by completing any system vocabulary word (which can be either a single word or a phrase) as follows (Figure 1):

1. *[Forward speech completion]*
   When the user prolongs a vowel in the middle of a system vocabulary word,[5] the system displays a numbered list of completion candidates whose beginnings acoustically resemble the uttered fragment.
   [Backward speech completion]
   When the user prolongs the last vowel of the wildcard keyword and then utters the last part of a system vocabulary word, the system displays a numbered list of candidates whose endings acoustically resemble the uttered last part.
2. The user can see other candidates by uttering the turning-the-page phrases, "next candidates" and "previous candidates," displayed whenever there are too many candidates to fit onto 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 saying (reading out) either its number, the rest of the word, or the entire word, that word is highlighted and used as the speech input result.

[3]This expression is very natural in Japanese.

[4]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.

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

4. SYSTEM DESCRIPTION

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 [3] that supports timestamp-based synchronization.

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 (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.

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 [4]. It is a bottom-up method that can detect an intentionally lengthened

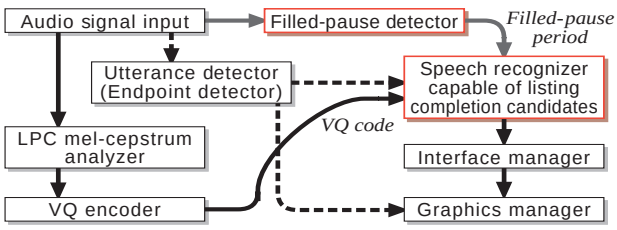

Fig. 2. *System architecture.*

vowel in any word without using top-down information (a 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. These features are found by using a sophisticated instantaneous-frequency-based analysis [4].

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.

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* [5]. In addition to a vocabulary of words to be input and be completed (e.g., the names of musicians and songs), this also uses a vocabulary for operating the interface (e.g., the wildcard keyword, the candidate numbers, and the turning-the-page phrases). All these words are stored in a tree structure as shown in Figure 3. The wedge marks in this figure represent multiple hypotheses maintained by a frame-synchronous Viterbi beam search decoder.

When the beginning of a filled pause is detected (about 200 ms after a vowel is lengthened), the recognizer enters the completion mode: it determines which completion method is to be invoked (forward or backward) on the basis of whether the wildcard keyword is the best hypothesis at that moment.

In forward speech completion, candidates are obtained by tracing from the top N_{seed} hypotheses (at the beginning of the pause) to the leaves (Figure 3): 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 3 is a seed, the completion candidates obtained are "Michael Bolton" and "Michael Jackson."

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 introduced an *entry node table* and the roots (nodes) from which the decoder starts searching are listed in that table. 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 3. Although a candidate can be selected by uttering just the last part, the recognition result sent to the interface manager is the entire word.

In backward speech completion, on the other hand, it is necessary to obtain completion candidates by recognizing a last-part fragment uttered after the wildcard keyword (after the end of the filled pause). We address this problem by temporarily adding to the table, just after the wildcard keyword, every syllable in the middle of all the vocabulary words (Figure 4). Then, after the last-part fragment is uttered, the hypotheses that have reached leaves are numbered in order of likelihood and the top N_{choice} hypotheses (leaves) are sent as completion candidates. After that, so that the user can select the correct candidate by reading out its first part, the recognizer temporarily adds an extra transition from the last node of the unuttered first part of each candidate to its leaf. For example, when the user utters *"nantoka– jakuson"* ("something–Jackson") for *"maikeru jakuson,"* ("Michael Jackson") an extra transition from the last node of *"maikeru"* to its leaf is temporarily added.

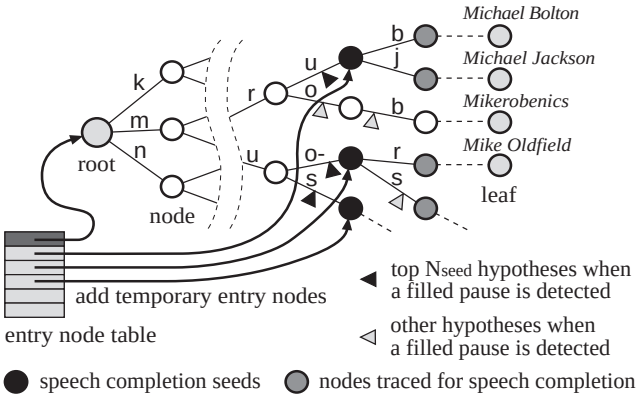

Fig. 3. *Forward speech completion: obtaining completion candidates on the vocabulary tree at the beginning of a filled pause and adding speech completion seeds to the entry node table.*

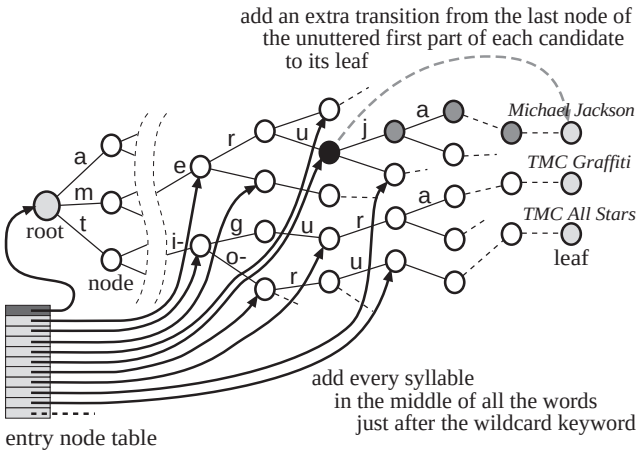

Fig. 4. *Backward speech completion: obtaining completion candidates by adding entry nodes just after the wildcard keyword.*

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

Figures 5 and 6 show examples of the graphics output of the implemented speech-completion system.[6] The current implementation uses the following parameter values: N_{choice} = 20 and N_{seed} = 15.

We tested this system with 45 Japanese subjects (24 male, 21 female). For the experiments, we used a system vocabulary comprising 521 entries (names of 179 Japanese musicians and 342 of their songs), which were collected from Japanese hit charts during fiscal 2000. To evaluate whether the subjects preferred to use speech completion, we measured the usage frequencies of speech completion under two conditions: (a) when a subject input a set of name entries from a list after gaining a good command of the speech-completion function, and (b) when a subject had to recall and input vaguely remembered entries. For condition (a), we had each subject input a set of five entries written on a paper sheet under the condition that the subject could freely use speech completion according to personal preference. Before the testing under this condition, the subjects input the same set with and without speech completion so that they would have a good command of speech

[6] We have also developed and demonstrated an application of this system, a speech-capable music jukebox system, which can play back a song whose title is determined through speech recognition with the speech-completion function.

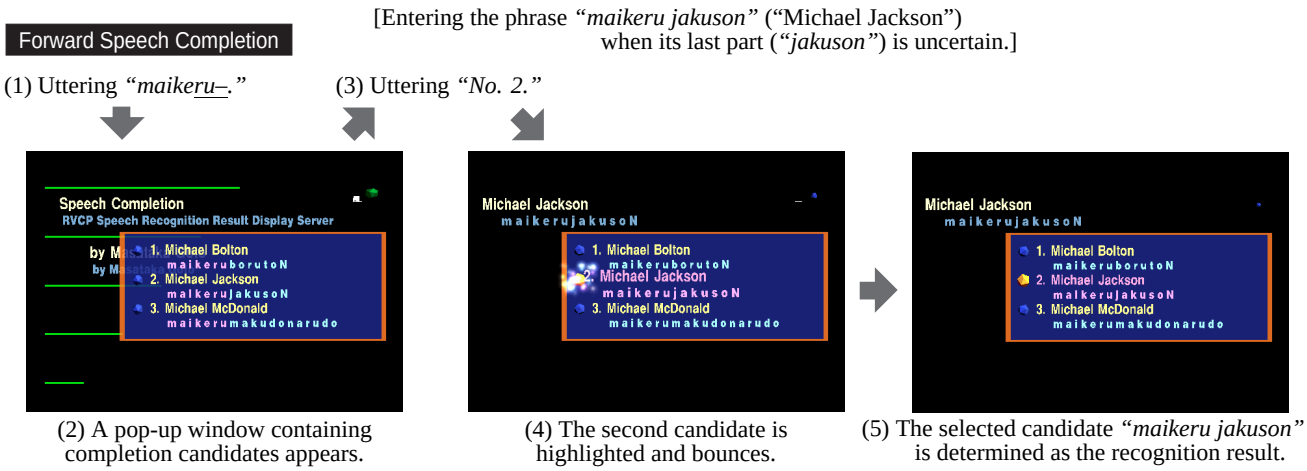

Fig. 5. *Screen snapshots of forward speech completion.*

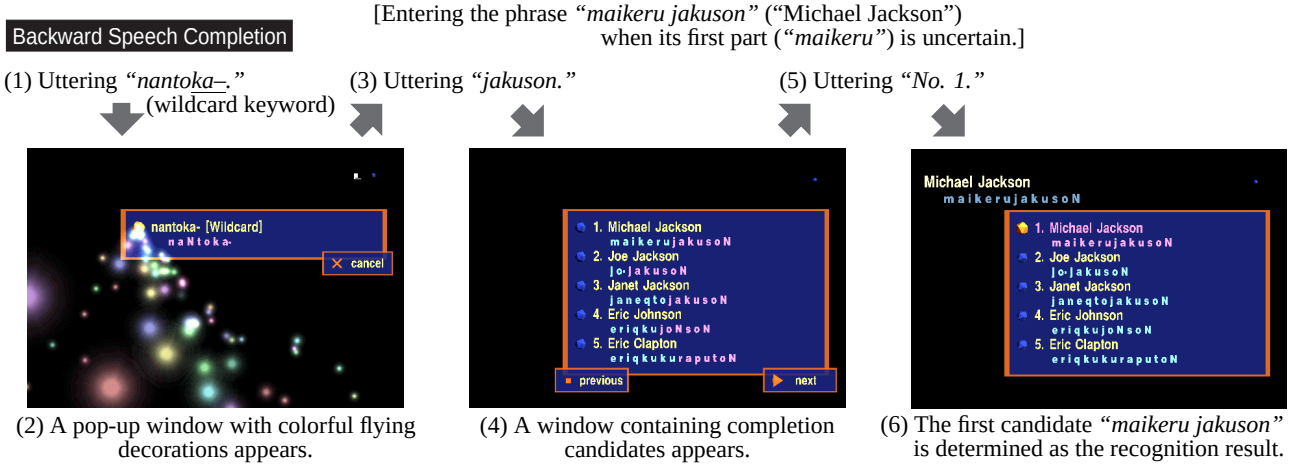

Fig. 6. *Screen snapshots of backward speech completion.*

completion. For condition (b), we took away the paper sheet listing the entries and had the subject recall and input as many entries as possible without being able to refer to the sheet. After testing under both conditions, the subject was asked to complete a subjective questionnaire.

We found that the average usage frequency of speech completion was 74.2% and 80.4%, respectively, for conditions (a) and (b). The relative usage frequencies for the two input methods (forward and backward) were 61.7% (forward) and 38.3% (backward) for condition (a) and 66.1% (forward) and 33.9% (backward) for condition (b). These results showed that the subjects preferred to use the speech-completion function even when they could choose not to use it. The questionnaire results indicated that the assistance provided by the listing of completion candidates was helpful and easy to use, that the speech completion made it easy to recall and input uncertain phrases, and that 80% of the subjects wanted to use the speech completion in the future.

6. CONCLUSION

We have described the new speech interface function *"speech completion,"* which fills in the missing part of a partially uttered fragment to help a user enter an uncertain phrase. To provide this completion assistance only when needed, we use a filled pause to invoke it. The filled pause is a good *"speech Tab"* trigger — an oral version of the completion-trigger key in text-based interfaces — because it can be detected independently of speech recognition and is naturally used in human-human speech communication. We have confirmed the effectiveness of this form of speech completion for the task of inputting the Japanese 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.

7. REFERENCES

[1] Toshiyuki Masui, "An efficient text input method for pen-based computers," in *Proc. of CHI'98*, 1998, pp. 328–335.

[2] Elizabeth Shriberg, "To 'errrr' is human: ecology and acoustics of speech disfluencies," *Journal of the International Phonetic Association*, vol. 31, no. 1, pp. 153–169, 2001.

[3] Masataka Goto, Ryo Neyama, and Yoichi Muraoka, "RMCP: Remote music control protocol — design and applications —," in *Proc. of Intl. Computer Music Conf.*, 1997, pp. 446–449.

[4] Masataka Goto, Katunobu Itou, and Satoru Hayamizu, "A real-time filled pause detection system for spontaneous speech recognition," in *Proc. of Eurospeech '99*, 1999, pp. 227–230.

[5] Katunobu Itou, Satoru Hayamizu, and Hozumi Tanaka, "Continuous speech recognition by context-dependent phonetic HMM and an efficient algorithm for finding N-best sentence hypotheses," in *Proc. of ICASSP 92*, 1992, pp. I–21–24.