INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 2, 155–164 (1997)

Turn-Taking as a Design Principle for Barge-In in Spoken Language Systems

REBECCA HEINS*, MARITA FRANZKE, MICHAEL DURIAN† AND ARUNA BAYYA‡
US WEST Advanced Technologies, 4001 Discovery Drive, Boulder, Colorado 80303
mfranzk@uswest.com

Received February 10, 1996; Accepted March 31, 1997

Abstract. It is widely acknowledged that users of Spoken Language Systems (SLS) want the ability to truncate system prompts by using a barge-in capability (e.g., Basson et al., 1995; Yankelovich et al., 1995). However, little has been published on how barge-in is used or if it adversely affects Automatic Speech Recognition (ASR) and the interface usability. Typically, user requests for barge-in are assumed to be based on the desire to make system interactions faster and therefore more similar to interactions with touch-tone systems. We believe that requests for a barge-in capability are rooted in the notion of discourse as a turn-taking event. Viewed in this way, we believe SLS can be enhanced to develop speech interfaces that are deemed more natural by users, as well as to increase system performance. This study addressed several issues. We found that users new to the system did not need to be informed about the barge-in capability before they attempted barge-in, that they used barge-in during almost half of their interactions with the system, and that they had identifiable patterns of barge-in use consistent with the turn-taking model. Results are presented and consequences for speech interface design as well as algorithm enhancement are discussed.

Keywords: spoken language systems, automatic speech recognition, barge-in, telephone interface, user interface design

Introduction

The study reported here was part of a larger research effort to develop a spoken language interface for Do Not Disturb™, a touch-tone-based call blocking service. Much of the prototyping effort centered around designing an interface that would permit the use of continuous speech rather than DTMF (dual tone multiple frequency, or touch-tone) input. However, this report focuses on subjects' use of barge-in, which was implemented in the last of several prototype designs. It has been widely argued that barge-in is a necessary component in systems incorporating Automatic Speech Recognition (ASR) (Basson et al., 1995; Yankelovich et al., 1995). Typically this is explained as a method of speeding up the task, or because it parallels touch-tone systems that permit DTMF input at any point during a system prompt. While we agree that these are important issues to users, we suggest that user requests for barge-in are an expression of the fundamental need to control and structure conversation, expressed in the notion of conversation as a turn-taking event (Sacks et al., 1975). We believe the turn-taking model can be used to enhance speech interface design. Firstly, knowing when a user is likely to interrupt will improve recognition accuracy through development of more intelligent algorithms. Secondly, providing a well-working barge-in capability will lead to the development of interfaces that feel natural to the user.

*Currently with Multicom Research.
†Presently with Pluto Technologies International.
‡Currently with Rockwell International.

The Turn-Taking Model

The model described in (Sacks et al., 1975), designates what constitutes a turn, when turn-taking is likely to occur, and how turn-taking is achieved. Briefly, according to this model a speaker is entitled to one turn-unit at a time at the end of which occurs a likely place to change speakers. The listener may 'self-select', or the speaker can convey to the listener that it is the listener's turn.

A turn-constructional unit is defined syntactically, i.e., sentence, clause, phrase, etc., such that it identifies for the listener what is being asked of them. For example, in the system we tested, a commonly played prompt was the confirmation prompt "To confirm say OKAY, to cancel say cancel." If the caller wanted to give an affirmative response, they would have all the information they needed in order to respond as soon as they heard the word 'OKAY'. In this scenario, the first half of our confirmation prompt constitutes a turn-constructional unit, and the phrase boundary a turn-relevance place. The speaker need not provide any additional information in order for the listener to be able to respond.

A listener will consider it relevant to take a turn as soon as he/she understands the speaker's request. In other words, whenever a listener has heard enough semantic information to make a decision about his or her next speech act. Significantly, turn-relevance places do not occur continuously throughout the speaker's discourse, but rather are discrete, recurring each time a turn-constructional unit can be completed by the listener. Phonological information such as intonation and pause also suggest to the listener when a turn-relevance place occurs. The naturally occurring pauses associated with many syntactic boundaries permit a listener to self-select without violating etiquette for interrupting. We believe this model can be used to design recognition algorithms that make predictions as to when to expect a response from the user, and given the timing, what the content is of their response.

Turn-allocation techniques fall into two general categories: current-speaker-selects-next-speaker, and next-speaker-self-selects. In the first case, a number of mechanisms may be used by the current speaker to select the next speaker. An obvious example of this is an addressed question such as "Will you make the call, Frank?" In this example, not only is Frank selected as the next speaker, the speaker has used a sequential unit, or adjacency pair, to constrain the next speaker's response. Frank is expected to be the next speaker and his response a yes or no answer. The same strategy is often incorporated in speech interface design. The interface must convey to the caller when it is their turn. Typically this is done by use of the pronoun 'you', or implicitly by the use of the imperative. The interface must also identify for the user what type of response it expects, given the limited vocabularies of recognition systems.

The second method of turn-allocation is next-speaker-self-selects. In this case, the listener takes the initiative to respond to something in the discourse. It is this case which we believe mandates the use of barge-in with automatic speech recognition systems. Speech interfaces that do not permit the use of barge-in deny users the ability to self-select. As soon as a user understands what is being asked of them by the system prompt, they want to be able to respond at the first transition relevance place. When instead, they are required to listen to a lengthy system prompt that provides them with additional information they consider irrelevant, they may become impatient, feel imposed upon, or become confused. Hence the feeling that the dialogue is not natural or that it is slow, with the resultant request for barge-in.

Overview

In the remainder of this paper, the interface prototype and the technology implementing it will be described. Then we describe the experimental method used in this study. In the results section, the system performance will be reported first. Then we address three questions with respect to the use of barge-in: Will subjects 'naturally' make use of barge-in, or must they be explicitly instructed in its use? How often will subjects make use of barge-in, and at which points are they likely to interrupt? Do they stick to 'turn-relevance places' as Sacks et al. (1995) suggest for natural conversation, or given that they are interacting with a machine, do they adopt less polite (and less predictable) ways of interrupting? Finally, we report on other observed consequences of introducing the barge-in capability. In the conclusion, the implications of these results for prompt design and algorithm enhancements will be discussed.

System Design

Speech Interface

The speech interface we tested was designed to make it faster and easier for subscribers to enter scheduling information when using their Do Not Disturb™ call screening service. Subscribers can personalize their service by scheduling when they want to accept calls and when calls are to be routed directly to their voice mail. Features such as recorded name, greeting, pre-scheduling and immediate access to call blocking are offered. The service also enables subscribers to give a passcode to privileged callers which allows them to "break through" the call blocking feature, even when calls are not generally being accepted. The current implementation of Do Not Disturb™ employs a DTMF interface which requires subscribers to enter all information, including day, time, and time-of-day information with (sometimes long) digit strings—a task that is neither intuitive nor expeditious. We wanted to find out if a continuous speech interface could be incorporated to increase the usability of the service. In addition to accommodating continuous speech input, the prototype we tested also had a barge-in capability, which allowed callers to interrupt the system prompts simply by speaking over them.

While current efforts in speech interface design often aim at making an interaction truly 'conversational' (Yankelovich et al., 1995; Aust et al., 1994), our prototypes were constrained by the attempt to make major parts of the speech interface parallel to the DTMF application, in order to make transitions between the two interaction modes possible. Implementation of a truly conversational interface would have also been difficult to achieve, given limitations of current ASR technology, and given that many of the call blocking service options are not naturally occuring tasks[1] so that relatively practiced and predictable, exchanges could not be expected. Indeed, new users may not even be aware of all of the options the service offers. The resulting prototypes were hybrid solutions, designed to permit speech input that had some flexibility without being completely unconstrained.

System prompts guided subjects through the interface by presenting a series of menu options. A keyword spotting algorithm provided subjects with the ability to respond with short phrases incorporating the keyword command. In addition, the interface included adaptive prompting, choice of prompt length, error correction protocols, and barge-in. Here is an interaction sequence, starting at the main menu prompt:

System: "Main Menu. Do Not Disturb is currently off. To turn it on now, say turn on. To work with your schedules, say schedules. To check emergency status, say emergency status, and for personal options, say personal options."
Subject: "Turn on till ten."
System: "I heard ten. If this is the time you wanted, please say AM or PM. To cancel, say cancel."
Subject: "Ten PM."
System: "Thank you. Do Not Disturb will be on until ten PM. To confirm, say OKAY. To cancel, say cancel."
Subject: "OKAY."
System: "Main Menu. Do Not Disturb is currently on. To turn it off now, say turn off. To work with your schedules, say schedules. To check emergency status, say emergency status. For personal options, say personal options."
Subject: "Emergency status, please."
System: "Emergency status is currently on. To turn it off now, say turn off. To go back, say go back."
Subject: "Go back."

Adaptive Prompting. Whenever time and day information had to be given, the system parsed the input and responded adaptively, enabling subjects to enter the necessary information in single or multiple units. In the dialogue above, the subject could have initially responded with "Turn on until ten PM". Alternatively, had they simply given the response "Turn on", the system would have prompted with "Please specify a time. Other choices are help and go back". The system was designed to parse the information it received and re-prompt accordingly, rather than requiring subjects to repeat information they had already given.

Prompt Length and Levels. In earlier prototypes we tested the effect of prompt length and level of explanation. We found that longer, more explanatory prompts lead to more consistent and less errorful user behavior, but lower user ratings. The current prototype had two prompt levels. Subjects were presented with the longer prompts by default, but had the option of switching to short prompts. The idea was to enable subjects to make the prompts less repetitious once they had become accustomed to the interface. Very few subjects opted to

switch to the shorter prompts in this study. In the sample interaction sequence above, the longer prompts are given. These findings are based on repeated use of prompts in single-session usability tests. Long-term user behavior might be different.

Error Correction. Error correction protocols varied according to the potential impact of the interaction. For all menu transitions and non-consequential steps, the call flow advanced without confirmation. The interface provided users with the ability to back out of an unwanted system state by saying "Go back". An explicit feedback loop was provided when scheduling information was being entered or the subject requested information be deleted. The system timed-out when speech was not detected within a set period of time. This would occur either when a user did not give a response or when no speech was detected. In these instances, a short reminder prompt was played listing the current options. This prompt also reminded users to interrupt the system at any time during the prompt.

ASR Technology

The prototype we tested consisted of a telephone-based speaker-independent continuous speech recognition component coupled with a speech detection component[2]. The call flow was configured as a finite state machine, with each state having a unique grammar. These grammars specified the vocabulary that was active for the recognition process of each finite state. The recognizer allowed flexible user input via a keyword spotting algorithm. In order to avoid erroneous speech detection induced by echoed system prompts (caused by signal reflections occurring at the telephone network switch), the speech detection component included an echo cancellation algorithm such that users were able to barge-in on a system prompt by speaking over it. The prototype operated in real time.

Callers engaged the prototype by dialing a number from an ordinary telephone. They heard a series of pre-recorded system prompts that offered choices for customizing their call screening service. Throughout the call, the speech detection component monitored the line for incoming speech. Callers made their selections simply by saying the command phrases into the phone.

Once speech was detected, the incoming speech was recorded and passed to the recognition component. Recognition was performed based on the active vocabulary, as defined by the grammar, for the given state of the call. The call flow advanced to the next finite state based on the output of the recognition component. At the time of this study, no provision for the rejection of speech input based on recognition accuracy had been implemented.

Methodology

Twenty native English speaking adults with unimpaired hearing were recruited for this study. Eleven men and nine women, balanced across the age groups of 26–35, 36–45, 46–55, and 56–65 were selected. None of the subjects had previous experience using speech recognition systems or Do Not Disturb. The sessions were conducted in a lab made to resemble a home living room setting. All of the sessions were videotaped, and an audio record of each subject response was made. Sessions ran from 1 to 1-1/2 hours in duration. Subjects were compensated for their time.

A very brief overview describing the service was read to each subject at the beginning of the session. Half of the subjects were told that the prototype system provided a barge-in mechanism, half were not. The subjects were then asked to perform a series of 18–20 tasks that manipulated all aspects of the system interface. A written scenario that attempted to simulate normal home use of the service was provided at the beginning of each task. Tasks included customizing their new service by recording their name and selecting a greeting, turning it on and off, and modifying the call screening feature to accommodate changes in their (simulated) daily routine. After each task, subjects rated the difficulty of their interaction with the system in brief questionnaires.

Some of the tasks were presented to the subjects more than once to allow us to measure subjects' learning performance. At the end of the session, subjects were asked to fill out a final questionnaire that asked them to describe their overall impression of the prototype.

The primary independent variable was knowledge of the use of barge-in. We knew that users wanted barge-in. However, we did not know how it would be used or if it would introduce any complications. We wanted to learn how much training information had to be provided to new users on the use of barge-in.

Secondary independent variables were age-group, gender, and computer experience. These variables were used to assess whether our interface design would be

usable to a variety of demographic groups, and not just by the young and technologically experienced.

The videotaped sessions were manually transcribed. Details concerning how and when subjects' responded, and the resultant action by the system, were compiled and analyzed.

Results

Our analyses found essentially no differences between groups based on the secondary independent variables of gender, age, and familiarity with high technology. While this result was somewhat unexpected, we take it to mean that the interface was unbiased towards any particular demographic group. However, as a consequence of these results, our findings are reported as overall averages across these variables unless otherwise specified.

System Performance

In order to accurately report on system performance we must break the results down into their component parts, as problems in one component can affect performance in another. To give an example, if the speech detection component only captured part of the input speech, the recognition would be on incomplete data, presumably with less than optimal accuracy. From the standpoint of the subject, the system may have failed to recognize what they said, but technically it was a problem within the speech detection component, not necessarily with recognition per se.

Speech Detection. Speech was properly detected with 90% accuracy, or in 1700 out of 1895 interactions (Fig. 1). Detection errors accounted for a total of 10% (195/1895) of the system interactions. Of these 195 detection errors, partial (incomplete) speech was detected 2% of the time (44/1895), and in 8% (151/1895) the speech detection component failed to detect any speech at all.

It should be noted that we distinguish between correct and incorrect cases of 'Nothing detected'. In the first case, subjects failed to provide a response and silence was correctly detected. No input was given to the recognition component, and call flow properly cycled into a re-prompting state. These occurrences have been included in the 'Correct detection' category of Fig. 1. When we report 'Nothing Detected' as an error condition (incorrect), subjects in fact gave a response which our algorithm failed to detect (151/1895).

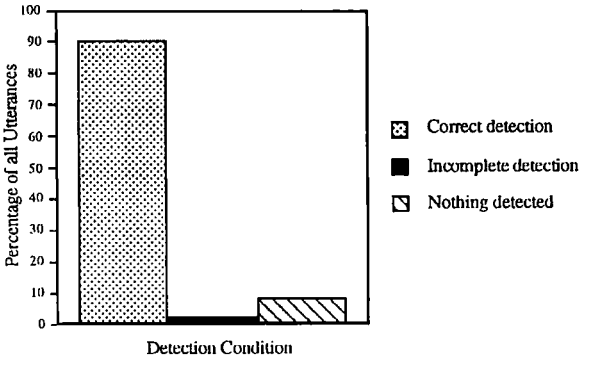

Figure 1. Endpoint detection performance.

It is evident from Fig. 1 that the endpoint detection component worked quite well. When there was an error, it was overwhelmingly one of failing to detect speech at all, rather than one of capturing an incomplete utterance. This is actually quite encouraging, as one of the parameters of the algorithm is a minimum detection threshold. We anticipate further testing will allow us to fine tune the level of the threshold and eliminate the majority of the problems in this category.

Recognition Accuracy. Not all of the system interactions reported above were passed to the recognition component. Some of the tasks in the study required subjects to record their name or input a telephone number using a touch-tone pad. In other cases, subjects failed to give a response, or our speech detection component failed to record any speech. None of these cases resulted in activity in the recognition component. We have also excluded the 26 out-of-vocabulary responses (see below) from our analysis. They do not contribute meaningfully to recognition accuracy, given the fact that the prototype lacked a rejection algorithm at the time of the study. With these omissions, we report on a total of 1646 recognition events.

Figure 2 compares recognition accuracy and error rates based on endpoint detection. When endpoint detection operated correctly, recognition accuracy was 94%, or correct in 1514 out of 1603 instances (left side of Fig. 2). Perhaps surprisingly, recognition accuracy fell only slightly, to 91% (39/43), when endpoint detection captured only partial utterances (right side of Fig. 2). However, as incomplete endpoint detection accounts for only 43 out of 1646 total recognition events (2%), overall recognition accuracy remains 94%. In

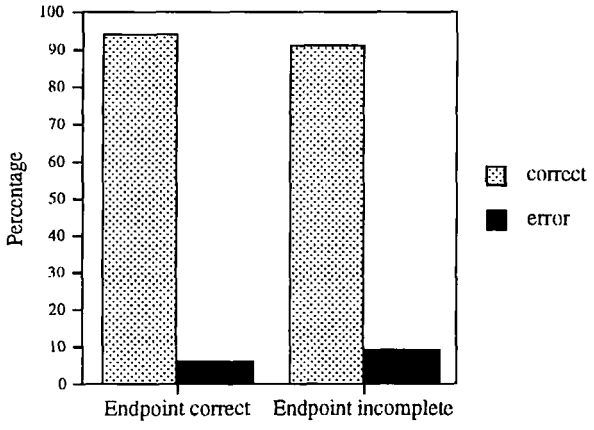

Figure 2. Recognition accuracy based on endpoint detection.

Section 5 we suggest methods for improving both endpoint detection, as well as recognition accuracy in systems with barge-in capability.

User Behavior

Use of Vocabulary. We use three categories to describe the types of input subjects used to respond to the system:

In-vocabulary:	an utterance that consists entirely and only of vocabulary specified in the grammar of a given finite state.
Out-of-vocabulary:	an utterance that does not contain any vocabulary specified in the grammar of a given finite state.
Keyword:	an utterance that contained one or more words of vocabulary specified in the grammar of a given finite state, in addition to extraneous speech.

Although our system was designed to accept continuous speech, 97% of the voice responses were in-vocabulary. We attribute this to the fact that the prompts explicitly told the subjects what to say, and the fact that as new users, they were unlikely to have preconceived ideas about what given input should be. Overwhelmingly, subjects repeated exactly what they had been told to say. The remaining 3% of the input speech were evenly divided between keyword and out-of-vocabulary responses. This means that the continuous speech portion of our algorithm was not effectively tested in this study. Further testing will be necessary to conclusively evaluate the performance of this component of the system.

Use of Barge-In. While it is generally agreed users want to be able to truncate system prompts via barge-in and there have been some reports on its usage frequency (Basson et al., 1995), references on the characteristics of barge-in use are scarce for SLS. We had several questions that we were trying to address with this study. The first question concerned training or teaching subjects about barge-in. Many previous studies have shown that linguistic behavior is usually altered when people talk to a machine (e.g., Hauptmann and Rudnicky, 1988; Franzke et al., 1993). Would users' natural tendencies to self-select in conversation carry over to a human-machine interaction in SLS[3]? To address this issue, half of our subjects (Group A) were explicitly informed about the barge-in capability during the introductory comments of their session. No mention of barge-in was made to the remaining subjects (Group B). Secondly, we wanted to know how much barge-in would be used, to determine whether or not the computational costs associated with providing this functionality were justified. Third, we wanted to identify patterns of use, if any. From a technical perspective, the system prompt could be barged-in on at any time. We wondered if Sack's notion of transition relevance place could be used to predict when barge-in was likely to occur, and if yes, whether or not we could use this information to enhance speech detection and recognition. Finally we wanted to ensure that the introduction of barge-in to an SLS interface did not adversely affect either user or system performance.

Is Explicit Instruction Necessary?

When we analyzed the data based on prior knowledge of barge-in, we found no statistical differences between the two groups as to their first attempt to barge-in on the system prompts ($t(1, 18) = .01$, $p > .99$). The means (Fig. 3) even show that subjects in group A (not told about barge-in) were slightly faster to use it, averaging just 3.5 interactions, that is 3.5 speech commands into the first task. Subjects in group B first barged in after 3.6 interactions on average.

These results confirm that new users of SLS will use a barge-in capability, and that little effort needs to be given to informing them of this feature. Subjects quickly got used to listening to the prompts and

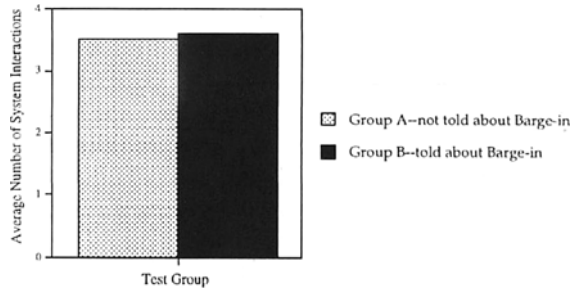

Figure 3. Average number of system interaction before first use of barge-in by subject groups.

interrupting when they had heard enough information to make a decision. Subjects did not have to be told that this option existed as our non-informed group (A) demonstrates. They quickly tried it, and when it worked they simply accepted it as an expected feature of a speech-based system.

How Often is Barge-In Used?

The group that had not been told about barge-in (A) used it for 42% of their utterances, on average. The group that had been told about barge-in (B) used it 50% of the time (see Fig. 4). This difference was not statistically significant ($t(1, 17) = -.75$, $p > .46$). Overall, barge-in was used in 46% of the interactions subjects had with the system.

This usage rate is in line with rates reported by Basson et al. (1995) for SLS implemented with keyword spotting technology. While barge-in was not used for every system interaction, subjects made use of it quite frequently. Our subjects were still 'novice' users of the system, even at the end of the trial. We would expect barge-in rates to increase as users become more experienced with a particular spoken language interface. However, many applications, such as those used to route customer service inquiries, will be used predominantly by callers unfamiliar with the system. Our findings have particular relevance to this type of application.

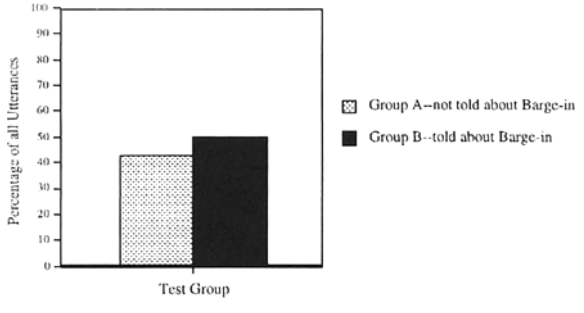

Figure 4. Use of barge-in across all system interactions.

Do Users Adhere to 'Turn-Relevance Points'?

We looked at the timing of speech input relative to the system prompts. The prompts essentially consisted of a series of keyword commands. We were interested in determining the relationship, if any, of the position of the keyword and the use of barge-in. We also wanted to know if the syntactic structure of the system prompt itself could be used to predict when subjects were most likely to respond. In some instances the keyword to be selected was necessarily in prompt-final position. In these cases, new users theoretically wouldn't have the information necessary to make a keyword choice until they had heard the entire prompt. We wondered if they would be more likely to forego the use of barge-in when the keyword command was prompt-final. For probability-based recognition algorithms, correlating timing of response with content of response could prove to be a powerful enhancement.

In order to capture the interaction of these factors, we categorized our data based on (a) whether or not the subject's keyword choice was prompt-final, (b) did they barge-in, and if so, (c) did the barge-in coincide with a syntactic boundary, (d) did they respond before they heard the keyword phrase or anytime after it, and (e) if after, did the subject respond immediately upon hearing the keyword spoken by the system prompt[4]. The cells in Table 1 show the interaction of these factors. The darkened cells in Table 1 represent interactions that are not logically possible. Examples[5] for the 10 relevant cells in this table are provided. In the examples, the barge-in point is marked with a "^". The numbers correlate with the cell numbers of Table 1. The italicized word is the keyword choice.

(1) "To confirm^, say *Okay*. To cancel say cancel."
(2) "To con^firm, say *Okay*. To cancel say cancel."
(3) "To confirm, say *Okay*. To cancel^ say cancel."
(4) "To confirm, say *Okay*. To cancel say can^cel."
(5) "To confirm, say *Okay*. To cancel say cancel."^
(6) "To confirm, say *Okay*^. To cancel say cancel."

The next set of examples demonstrate the cases when the keyword choice occurred prompt-final. Again, the example number correlates to the cell number of

Table 1. Distribution of user utterances across all logically possible points in the prompts.

	Keyword not final				Keyword final			
	Barge-in		No barge-in		Barge-in		No barge-in	
	Boundary	No boundary	Boundary	No boundary	Boundary	No boundary	Boundary	No boundary
Before keyword	(1) 0	(2) 0			(7) 0	(8) 0		
After keyword	(3) 788	(4) 84	(5) 759				(9) 264	
Immediately after	(6) (589)						(10) (264)	

Table 1, the "^" denotes the barge-in point, and the keyword choice is in italics.

(7) "To confirm^, say Okay. To cancel say *cancel*."
(8) "To con^firm, say Okay. To cancel say *cancel*."
(9) "To confirm, say Okay. To cancel say *cancel*."^
(10) "To confirm, say Okay. To cancel say *cancel*."^

If users did not have any type of rules for timing their response, we would expect an even distribution of data points in each of the logically possible cells. If the request for a barge-in capability were based solely on speeding up system interactions, we would expect some interruptions to occur before users heard the keyword in familiar prompts such as the confirmation prompt. In this case, we would expect to see data reported in cells 1, 2, 7, or 8. Or, if SLS users are not sensitive to syntactic structure in timing their response, we would expect to see some accretion of cases in cells 2, 4, and 8.

None of these predictions are supported by the data in Table 1. The distribution of barge-in cases across the cells is far from random, indicating SLS users do have some type of response strategy they are using. We expected subjects to be aggressive barge-in users, responding before the keyword phrase, however this hypothesis was disproved as well. This may not seem so surprising, given that our subjects were all new users to the system. However, on average subjects heard the confirmation prompt ('To confirm say OKAY, to cancel say cancel.') sixteen times throughout their sessions—often enough for even new users to be familiar with it. We'd like to emphasize that the system continuously supported barge-in. From a technological point of view, subjects had the ability to say their keyword choice at anytime, including *before* the prompt listed it as an option. Nonetheless, in our study, subjects never did this. The prevailing strategy was to respond only after hearing their keyword choice.

If one considers the theory of turn-taking that we briefly outlined in the introduction, the pattern of results in Table 1 is not surprising at all. Sacks et al. (1975) predict that discourse participants tend to respond in predictable places, based on turn-constructional units and transition relevance places. Speakers seem to wait until they have enough semantic information to make an informed decision. Once they have made a decision, they wait until the next phrase boundary allows them to interrupt. This tendency is so strong that our participants did not stray from it once, despite repeated exposures to the appropriate keyword.

The data in Table 1 also confirm that subjects tended to interrupt more at syntactic boundaries, rather than in the middle of a syntactic constituent. Nine times out of ten (788/872), the barge-in event occurred at a syntactic boundary. Of these, 75% (589/788) immediately followed the keyword. This is more evidence for Sacks' notion of speakers and listeners sensitivity to turn-relevance places. When listeners self-select, they will wait until they have all necessary information to take their turn, but they tend to use the next possible syntactic boundary to break in. The fact that our participants self-selected in the middle of a syntactic constituent (cell 4) roughly 10% of the time, indicates that this is a strong, but not binding constraint. As with most other discourse issues, iron-clad rules are not the norm.

Does Barge-In Adversely Affect User Behavior or System Performance?

As we analyzed our data, we were sensitive to any trends that would indicate that the inclusion of barge-in had introduced unexpected (and undesirable) consequences for our subjects and for system performance. We report on two of our findings here.

As part of our study, subjects were asked to complete some of the tasks more than once. We designed the study this way so that we could measure the learnability of the system. One measure of learnability is the number of interactions subjects used to complete a particular task. On a repeated task, more interactions would indicate that subjects either had trouble navigating the interface, or that it was somehow confusing to them, causing them to take unnecessary actions to complete the assigned task. Fewer interactions would suggest that the interface was easy to learn, and that subjects quickly conceptualized the menu structure and the prompting style.

We found that on repeated tasks, subjects tended to have fewer system interactions[6]. However, some subjects got lost in the menu structure when they were repeating a task, causing them to back out of an unwanted system state, listen to the choices again, and select the correct menu option. Several of these subjects self-reported that they had barged-in before listening to all of the menu options, falsely assuming they had enough familiarity with the system to make their selection before hearing all of the options. In other words, the inclusion of barge-in resulted in increased confusability by allowing subjects the option of responding before hearing all of the options available to them.

A final minor point concerns disfluent speech input. We use the term 'disfluency' as a cover term to include false starts, hesitations, incomplete utterances and repetitions in the subjects' responses. We noted that while there were very few disfluencies overall, only 2% of total inputs, of these 91% (32/35) coincided with barge-in. This is significant since they may cause higher recognition error rates. System developers should anticipate higher rates of disfluent speech in conjunction with barge-in, and modify systems accordingly.

Discussion

While we readily admit that the barge-in usage characteristics reported in this paper may not necessarily be generalized across all users of all SLS, we believe they are important findings nonetheless. All SLS will have some percentage of new users, and some, such as customer service applications, may have almost exclusively new users. We are also persuaded that different prompting styles will invariably affect barge-in use. However, menu-based interfaces in which users are requested to say keywords result in high task completion by users, and many applications will no doubt continue to be developed using this prompting style.

The results presented above unambiguously support our claim that user requests for barge-in are more deeply rooted than the desire to speed up system interactions or mimic familiar touch-tone interfaces. Overwhelmingly, our subjects showed a strong tendency to adhere to the turn-taking model developed by Sacks et al. (1975), even though they were interacting with a machine. Our results confirm a general tendency to self-select at the first turn relevance place following the end of a turn-constructional unit, even when subjects are conversing with an SLS system. We hope these findings will result in the enhancement of applications incorporating speech technology.

We are convinced that the inclusion of barge-in technology is well worth the computational costs associated with it. The usage rates reported for this study suggest little or no effort need be expended on informing potential system users about barge-in. Users readily barge-in, apparently assuming it is a standard feature of SLS. We believe inclusion of barge-in will contribute to overall user satisfaction with speech interfaces, not only because it speeds up system interactions, but because it allows for a more natural discourse as well.

Inclusion of barge-in technology will necessarily impact interface design strategies. Interfaces based on keyword spotting technologies must provide unambiguous command choices, to prevent users from 'second guessing' the system and ending up in an unwanted system state. A balance has to be found between the increased flexibility barge-in allows and the potential impact on users' ability to navigate the system as a whole.

The findings reported here also have relevance to speech detection and recognition algorithm development. In applications geared to new users, probabilistic recognition systems should be able to incorporate user response strategies to improve both speech detection and recognition accuracy. Rather than having a fixed parameter which sets the threshold for speech

detection, a dynamic system coordinated with the syntactic structure of the system prompts could be implemented. We propose a slightly higher threshold be used when input speech is not predicted, making it less likely that background noise will be falsely detected as speech input. A more sensitive threshold could be activated when a response is predicted to occur, such as immediately following a keyword, making it more likely that speech input will be properly detected.

By correlating the timing of a user's response with information about which keywords they heard, we believe probabilistic recognizers could incorporate a secondary, semantic comparison resulting in improved recognition accuracy. Essentially, we propose development of a system capable of predicting *what* was said, based on *when* it was said. In those instances when the entire prompt was played, the 'grammar' of this proposed semantic component could be equally weighted, or weighted to favor the prompt-final keyword. For barge-in events, the grammar could be weighted to favor keywords played before barge-in, with the last keyword played out weighted most heavily of all. By keeping track of the timing of speech input relative to the keywords played out by the system prompt, the N-best choice from the recognition component could be compared against the N-best semantic choice. We believe this would result in a more intelligent system, with favorable impact on system performance.

Future Research

The results reported here represent a first analysis of barge-in use in spoken language systems incorporating ASR. Many areas for continued research remain. We would expect barge-in use to change as users become more familiar with individual applications. Understanding this change will be very important for SLS interface design. The effects of overall prompt length may also be key to predicting barge-in use (as well as user satisfaction with SLS interfaces), and the effect of prompting style, i.e., conversational vs. menu-based systems, may also play an important role.

Notes

1. The scheduling component is a notable exception. Earlier in the prototype development cycle, we conducted extensive Wizard-of-Oz studies to identify how people talk about scheduling. We used the results of these studies to design the scheduling component of the speech interface.
2. The speech recognition component was co-developed by US WEST Advanced Technologies and the Oregon Graduate Institute. The speech detection component was developed by US WEST Advanced Technologies.
3. Stuart et al. (1991) report on prompt truncation in DTMF systems. However, as Marx and Phillips (1995) point out, assumptions regarding DTMF systems (by both users and system designers) do not necessarily carry over to systems incorporating ASR.
4. For the purpose of this study, we did not evaluate the interaction of overall prompt length with barge-in, or the use of barge-in relative to individual prompts. Instead, the experiment was designed to expose subjects to virtually all of the system's capabilities. Our data therefore reflect user interaction across the system prompts as a whole.
5. In some cases, there are multiple instances at which a subject could have responded that fit the category defined by a given cell. Our examples are not intended to be comprehensive. Rather, we are providing a single example of each type to illustrate each cell in Table 1.
6. It must be noted that subjects were not specifically asked to complete tasks as quickly and efficiently as possible. This precludes making statistical claims, as some subjects showed genuine interest in learning about the service. We observed some subjects selecting a menu option because they were curious to learn more about it, not because it contributed to efficient task completion or because they were lost or confused.

References

Aust, H., Oerder, M., Seide, F., and Steinbiss, V. (1994). Experience with the Phillips automatic train timetable information system. *Proc. IEEE Workshop on Interactive Voice Technology for Telecommunications Applications*. New York: IEEE Press, pp. 67–72.

Basson, S., Kalyanswamy, A., Man, E., Springer, S., and Yashchin, D. (1995). Establishing speech technology requirements: The *money talks* field trial. *Proc. Annual International Voice Technologies Applications*. San Jose, California: American Voice Input/Output Society, pp. 131–136.

Franzke, M., Marx, A.N., Roberts, T.L., and Engelbeck, G.E. (1993). Is Speech Recognition Usable? An exploration of the usability of a speech-based voice mail interface. *SIGCHI Bulletin*, 25:49–51. New York: Association for Computing Machinery Inc.

Marx, M. and Phillips, M. (1995). Against "Shoehorning:" Rethinking IVR architectures for speech recognition. *Proc. Annual International Voice Technologies Applications*. San Jose, California: American Voice Input/Output Society, pp. 187–195.

Rudnicky, A.I. and Hauptmann, A.G. (1988). Talking to computers: An empirical investigation. *International Journal of Man-Machine Studies*, *28*:583–604.

Sacks, H., Schegloff, E., and Jefferson, G. (1975). A simplest systematics for the organization of turn-taking for conversation. *Language*, *50*:696–735. Washington, D.C.: Linguistic Society of America.

Stuart, R., Desurvire, H., and Dews, S. (1991). The truncation of prompts in phone based interfaces: Using TOTT in evaluations. *Proc. of the Human Factors Society 35th Annual Meeting*. Santa Monica, CA: Human Factors Society, pp. 230–234.

Yankelovich, N., Levow, G., and Marx, M. (1995). Designing speech acts: Issues in speech user interfaces. *SIGCHI, Human Factors in Computing System Proc., Annual Conference Series*. New York: Association for Computing Machinery Inc., pp. 369–376.