Conversational Assistant for In-car Task Management

Joseph Reisinger, Mahesh Viswanathan and **Liam Comerford**
{jreisin, maheshv, lcmrfrd}@us.ibm.com
IBM TJ Watson Research Center
1101 Kitchawan Road-Route 134
Yorktown Heights, NY 10598

Abstract

The number of devices (in-car navigation systems, cell phones, email, web services, etc.) competing for a driver's attention is increasing creating the need for controlling cognitive load. We present a system, the Persistent Conversational Assistant, which strives to reduce cognitive load by providing the user with an intelligent-seeming assistant. Intelligence on the assistant's part can reduce the need for the driver to adapt to the system interactions. There are significant psychological factors to be addressed in building such a system. Since the in-car domain generally involves only a few users interacting with the system over a large period of time, the "bank teller" or "phone menu" style of interaction, though easier to design and implement, may actually increase user frustration, reducing the user's cognitive capacity. We posit that it is better to provide a "persistent assistant" that learns the drivers preferences and proactively simplifies common tasks.

Introduction

Auto manufacturers are building next-generation vehicles with new devices that provide local and connected services allowing users to access rich information resources such as driving directions, points of interest, email, calendar, traffic alerts, music, news, emergency assistance, accident avoidance systems, etc. We are building a prototype of a conversational agent, the Persistent Conversational Assistant (PCA), for automotive telematics that can access and operate such applications and services by conversing with its users. The PCA effort is framed in the automotive domain but its design is scalable and portable to other non-automotive environments.

The proliferation of devices in the car increases the need to make each of the devices easy to use with minimum distraction to the driver. One can define usage difficulty as requiring the user to look away from the road or remove a hand from the wheel. The solution, eyes-free hands-free control of some of the driving environment, can be obtained by applying appropriate speech technologies to access and control certain resources. An assistant that provides a unified consistent interface to all the devices and applications (pre-installed, third party, or driver installed) can satisfy the easy-to-use requirement. An in-car agent that can adapt and demonstrate autonomous behavior by warning driver of hazards such as approaching vehicles, slipping tires, or over-speeding, can make a safer driving environment.

The most prevalent speech interface designs use the so-called "bank teller" model. The bank teller fulfills a specialized task for which it is designed to be an expert. It asks pertinent, domain-specific questions in order to complete tasks organized along the lines of queries posed by a bank's agent to fill out an application form for a loan or mortgage. Digressions from the form can be detected but the tolerance is often limited because the bank teller often exhibits expertise on just one concept at a time. These systems can be experts and may be able to handle dialogs of some complexity, but ultimately they are domain specialists.

An alternative model is the "butler" model. The butler is expected to be informed, helpful, resourceful and capable of learning from the user and his usage patterns. In addition, the butler serves the user's needs by providing a single consistent interface. It avoids unexpected behavior and thus reduces the planning requirements reflected on the user. It monitors user input, directs it to the appropriate specialist, and redirects responses back to the user, while mindful that the user does not know or wish to know how that information is gathered. In particular, the butler must:

- permit intuitive free-form request and commands and not require that the user commit commands to memory,
- have a single integrated end-user interface to all devices whether built-in, the users or some third party's,
- be capable of interacting intelligently with the user by providing information taking into account the driver's cognitive load or degree of familiarity with the system,
- be capable of substantial variability in output speech wording, inflection, and information detail,
- learn about the user by abstracting simple usage patterns, and then allowing execution of a set of simple commands with a voice macro; or have different levels of use (novice to expert) depending on users degree of comfort with the conversational interface,
- be configurable by the car manufacturer - for setting broad guidelines for use of certain services,

- be configurable by the end user - for prompt levels, user name, voicing, and other personalization features,
- retain the context of users input and respond accordingly, potentially across driving episodes,
- be able to take supervisory instructions from entities such as a workload manager (an engine that monitors the car's internals and provides alerts should something potentially hazardous occur) and act accordingly, and
- be able to integrate gestures and voice requests and commands and respond accordingly.

The PCA user interface design is influenced by three factors: the number of varied devices and services currently and soon to be available, the requirement that each of these devices and services be easy to use, and that the user be safer operating the same set of car services through the PCA than through manual controls. The opportunities for distraction increase as the number of in-vehicle devices and services increase. If the user must divert a significant amount of attention in order to operate some new feature, it is reasonable to believe that the safety of the driver (and occupants) is diminished. If the attention that the user assumes can be spared to operate a feature is more than makes them comfortable, ease-of-use is diminished. Synergistic effects between services are subject to the same problems. Our expectation is that merging control of non-critical services into an intelligent conversational speech interface that one can address in simple, natural language can successfully address this problem. Such an interface may be designed to be "aware" of the drivers driving habits and cognitive load, the cars situation (its internal and external environment), and use this information to gate the demands of different devices and services for the users attention. A further (long-term) goal of this project at IBM is to build a single interface to all devices that integrates voice and gestures so that the bank-teller like properties of automotive GUIs (graphical user interface) can be mitigated.

This paper outlines some past work done in reducing the drivers cognitive load, and sets forth the hypothesis that a command and control multimodal interface with a primary speech component is not sufficient for accomplishing this reduction. Rather, the psychological factors at play in long-term interaction and partnership must be examined. Similarly, underlying AI implementation issues must be examined.

Reducing Cognitive Load

One of the perceived causes of automobile accidents is *driver distraction*, which can be due to factors external to the car (e.g., beautiful scenery), in-car factors external to the user (e.g., cell phones, pagers, etc.), or driver-related factors (need for rest, inebriation, inexperience, illness). Distraction is problematic because it reduces a driver's *cognitive capacity* available to the task of driving to managing. From a high-level cognitive perspective, the driver has limited cognitive resources and must allocate an appropriate percentage of that capacity in order to drive safely. As distractions impose on demands on cognitive resources, the driver's ability to react to events outside the car safely becomes degraded.

Several studies have been conducted which indicate a correlation between cell phone use and accident risk (Redelmeier & Tibshirani 1997; Violanti 1998). Furthermore, there is little apparent difference between accident risk for short calls, and accident risk for longer, indicating that the act of conversation itself (not just the the act of picking up the phone) is a factor (Curry 2001). Practically, however, if devices are going to be used while driving, the least distracting interfaces possible should be presented to the driver (McCallum *et al.* 2004).

Wickens' *Multiple resource theory* suggests that two concurrent tasks with different modalities would have less interference than two tasks with the same modality (Wickens 1984). In other words, combining a manual-visual task and a speech task would result in less cognitive load than two concurrent visual tasks. Studies done by Gellantly and Dingus (Gellantly & Dingus 1998), Lee et. al. (Lee *et al.* 2001), and a group at Ford Motor Company (Greenberg *et al.* 2003) tend to confirm this hypothesis.

Recent driving simulation work by McCallum and Brown also supports these findings (McCallum *et al.* 2004). Their work makes use of the *Battelle In-Vehicle Telematics Research Environment*, which simulates roadway condition and in-vehicle device information in order to test drivers in a variety of driving situations. In an experiment requiring drivers to operate a PDA to perform routine tasks (phone dialing, email, etc.), McCallum and Brown found that a PDA with a manual touchpad interface lowered drivers' mean reaction time by 0.44 seconds, while a PDA with a speech interface lowered the reaction time by 0.21 seconds. The difference between the baseline no-PDA performance and the performance using the speech interface did not differ by a statistically significant margin. Based on this data, McCallum and Brown suggest that in-car systems with minimally distracting interfaces should "result in reduced accidents and injuries, as well as user acceptance" (McCallum *et al.* 2004). They go on to point out, however, that factors such as recognition accuracy and response latency on the part of the speech system are key factors determining how distracting the interface is.

Since user acceptance is required for a speech interface system to be effective, in addition to pure computational factors (e.g., speech recognition rate), some HCI and AI factors must be taken into account to make the drivers use of the system as smooth as possible. For example, as the work of Clifford Nass and his collaborators indicates, subtle elements such as personality cues in text-to-speech (TTS) synthesis affect how much the user trusts the system (Moon & Nass 1996; Nass & Lee 2000). It is our position that command and control speech interfaces alone will not sufficiently solve the problem of reducing cognitive load. Instead, work needs to be done making the interface intelligent, personable, adaptable, and effective. This ultimately requires the system to act as a personal assistant, and actively learn the best interaction styles for a specific user.

Improving User Experience

There are two classes of limitations inherent in spoken language dialog systems: *HCI limitations*, and *technological limitations*. An HCI limitation exists when the user is unable to express their intent using the affordances of the interface or the interface is unable to discern the user's intent from their actions. This may be due to shortcomings in the user, the interface, or the processing of user-generated events. Technological limitations comprise the set of post-interface processes which are needed for fully "human" interaction but which are currently unsolved problems. Both forms of limitation increase the user's cognitive burden. In one case finding expression requires effort and attention, and in the other, avoiding placing impossible demands on the system requires effort and attention.

HCI limitations include all deficiencies in the user-interface that cause user frustration. Broadly speaking, these are deficiencies in the completeness with which the interface models human conversation. For example, requiring the user to perform the same repetitious tasks often, or not accurately keeping track of the user's perceived dialog context so that, for example, anaphora can be resolved correctly. These limitations may arise from the user misjudging the capabilities of the system, and are, for the most part, addressable within the scope of a suitably powerful dialog management system.

Technological limitations are not addressable beyond incremental research and hardware advancement. These limitations include things like speech recognition rates, or text-to-speech fidelity, correct handling of user sarcasm, etc. In this paper, these limitations are not addressed as research problems; instead a system is proposed which will function properly within a certain error bound for speech recognition.

HCI Limitations

Much user frustration aimed at dialog systems comes from having to perform the same or similar dialog turns or having to listen to the same generic prompts more than once in a conversation. This is particularly true in "bank teller" style interactions, where little emphasis is placed on making the speech interface anything other than a means to complete a predefined set of tasks. In any case, this model is sufficient for automated call-centers, where user interaction occurs in short, infrequent, goal oriented dialogs. It is expected that most users would reject such a system after a few hours of interaction in a car. Since the system is not learning, nor making any effort to simplify common dialog functions (as any human assistant would do), the user will quickly become frustrated and cease interaction.

Since the primary interface to the assistant is speech, the user will mostly draw experience with past automated voice systems, and from past *human-human conversations* (Reeves & Nass 1996; Shechtman & Horowitz 2003). In the lack of any past experience with computerized voice systems, the user will default to common sense knowledge for interacting with other humans. A maximally facilitating assistant can capitalize on common sense knowledge if it understands a large degree of conversation styles. Current dialog systems generally do not take advantage of the user's human-human conversation knowledge.

The topics that must be addressed in order to build a consistent, human-like interface include:

- *System personality* is generally not implemented on a global level beyond keeping the system's utterances polite and concise. Since individual users may prefer a more or less proactive system, or more or less polite system depending on their tastes, the personality should be easily mutable, perhaps even automatically so. Consistent personalities are considered to be the norm in humans, so it is reasonable to assume that a consistent system personality will be less abrasive to most users.
- *Contextual understanding* is critical in implementing a human-like system. During a dialog, a data structure storing *context* should be maintained in order to resolve various types of anaphora (e.g., "Turn it up some more.").
- *System learning* - The system should not force the user to learn a particular interaction style for task completion, and should instead rely on its own learning to understand the user's goal. For example if the user prefers saying "Plot a route to Yorktown" to mean "Plot a route to the IBM TJ Watson Research Center," then the system should detect this, and add it to its grammar. On a larger scale learning can include determining the user's preferred interaction style (and skill level) and adapting the dialogs based on that level.
- Finally, *robust dialog failure recovery* is necessary to compensate for the natural computational limitations of the system. The system should be able to gracefully recover by offering alternative dialog paths when the user makes it clear that his intentions are not being met.

"How may I help you?" style impersonal clerks are fine for telephony systems (i.e., short duration, task-directed interaction), however, we contend that they pose as an irritant to users who must interact with the system over the long-term.

Computational Limitations

The reality of implementing a conversational dialog system in the car must be taken into account at all phases of design, since the primary source of user irritation will be in cleaning up after the system mis-recognizes an utterance. There are two places a misunderstanding between the car and the driver can occur: during transcription, and during interpretation.

- On the *transcription level*, errors can occur due to noise in the environment (e.g. engine, wind, passengers, etc.), or due to limitations in the speech recognition engine's models which prevent the system from capturing certain dialects or words easily. Environmental noise is particularly a problem when the car is in motion, and can vary depending on the speed. One way to protect against this is to switch the recognition engine models based on the speed of the car.
- The other source of recognition error is on the *interpretation level*. Even if the user's utterance is completely

understood, the natural language understanding components of the system may not translate the utterance into the correct application context. This is even more common when the user utterance contains little information and relies heavily on previously constructed dialog context (e.g., "Do it again"). Interpretation level error also covers the case where the system does not have the appropriate information to deal with the utterance, but executes a command anyway (i.e., the user makes a request for non-existent functionality).

Outside of recognition error, other computational limitations that affect the user's satisfaction with the system are TTS voice quality (inconsistencies in pronunciation or lack of inflection) and recognition delay (sometimes several seconds to decode and process an utterance). Addressing these limitations directly is outside of the scope of this paper, rather the proposed dialog management system is architected specifically to mitigate the irritation caused by these limitations.

Dialog Management Systems

A Conversational Resource Manager (CRM, originally the Embedded Dialog Manager (Comerford *et al.* 2001)), provides a simple, modular framework for implementing command-and-control oriented personal-assistant speech systems, such as the PCA. The CRM has many responsibilities including scheduling, loading services, resource allocation, interprocess communication in addition to presenting a single, consistent, user interface to the user. The CRM is based on rich user interface descriptive data, message queuing and a priority system which maintains a dialog flow consistent with the users expectations. It is also is capable of handling critical interruptions and dialog forking as sources of data come on-line or go off-line or the user pursues a digression. The CRM supports complex features such as context awareness, dialog breakdown detection, user modeling, and learning. Further, user interface description mechanisms in the CRM are rich enough to support limited anaphora resolution.

The CRM framework is designed to support four levels of interaction: reflexive, time-binding, inference and learning, and autonomy:

- *Reflexive* interaction is command based: e.g., "Show me the map" or "Plot a route to San Jose." With reflexive interactions there is little sense of context and the system has no autonomy beyond responding to dialog breakdown indicators or critical interruptions.
- *Time-binding* interaction maintains a sense of dialog context, and is capable of switching seamlessly between tasks. An example is the user partially completing a navigation task, but suspending it to check an email message.
- *Inference and learning* enabled interaction allows the system to adapt to the user's preferred interaction style based on analyzing past interaction data. For example if the user prefers a certain radio station, the system can offer to to store it as a preset.
- *Autonomy* is the highest level of interaction, where the system automatically makes interface and functionality changes based on observed behavior, with little or no prompting. This level of interaction requires a high-degree of trust between the user and the system.

Firm support for reflexive and time-binding interactions currently exist with research being directed at inference, learning, and autonomy.

The PCA consists of the CRM, user interface data consisting of the input "vocabulary" (i.e., voice, haptic, etc.) that may be used, and output system response (i.e., text for audio generation, haptic, graphical, etc.), specialized services (i.e., applications, engines, the internet) to handle the user's domain-specific requests, gesture or voice recognition engine to handle user input, and output engines to represent the system response. The evolution of dialog managers and subsequent necessity for including these features will be discussed in the next section.

Design Evolution

Voice user interfaces have gone through evolutionary iterations including the *application* model, *dialog manager* (DM) model, and the *conversational* model. In the application model, speech engines are directly linked to applications as services to the application. Applications address the speech engines for both input and output through an API and receive the provided services (e.g., decoding audio into text or transforming text into audio). Application writers are responsible for coding the dialog interface, thus it is termed "application centric." The user is provided with a fixed vocabulary of words or phrases that when spoken amount to pushing (potentially, a large number of) invisible buttons. Prompting the user on which buttons are available is usually accomplished through a simple spoken list, reminiscent of call-center automation systems. Such systems rarely implement interfaces the exceed the "reflex" level of command and control.

The next evolution in speech systems is the DM-centric application. In these applications, both speech engines and specialized applications provide services to a dialog manager. This DM is responsible for discerning and guiding completion of the "task" the user is attempting to perform (e.g., route plotting or address-finding) through to task completion using pre-scripted dialogs. In this sense, the DM is responsible for generating the Voice User Interface (VUI). The application writer is responsible for providing the API to the application services. A GUI may still be managed by the application, but the requirement that the application provide support for managing the engine requirements (e.g., initializations, callbacks, engine control, vocabulary change, etc.) has been removed. Furthermore, the VUI interface is specified to the DM in the form of data or data mixed with code and is presented as tasks or subtasks. This methodology has two substantial advantages: division of labor and code stability. In HCI terms, it also improves the VUI consistency.

Software that uses a DM adds a human property to the reflexive capabilities of the application model system. This is

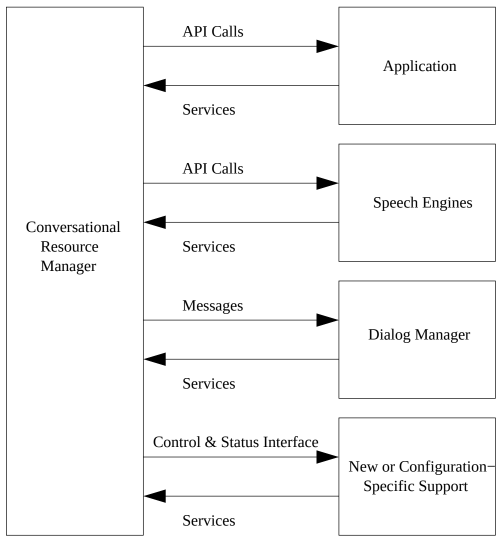

Figure 1: In the Conversational Resource Manager (CRM) model, the architecture is no longer dialog manager centric. Instead, DM's can be added to the system depending on the application.

the property of "time binding." The control window extends past the most recently uttered phrase. Thus a user can say phrases like "send this email to Jack" and then later, "also send it to Fred," and expect the system to understand that the "it" in the second phrase refers to "this email" in the first. Similarly, directed dialog (style of dialog where the system poses very specific questions requiring short answers) for task completion is time binding. In other words, the system may realize that it has insufficient information on which to act, and thus prompt the user to provide the missing information via a set of carefully crafted questions.

By applying the idea of division of labor and increasing approximation of human properties, the DM model extends to more complex functionality. The properties of such an extended software system, while incorporating reflexive, and time binding capabilities, could be expanded to include, at least, inference and learning (and perhaps permit some limited autonomy) without fundamental changes to the underlying architecture.

Conversational Resource Manager

One of the main concerns of car manufacturers is whether a user's experience will be satisfactory. Given the sustained contact time between the in-car system and its users, this concern is legitimate. Highly specialized dialog systems (e.g., telephone, kiosk or the web-based systems) are designed with task completion as the principal goal. In other words these systems are designed for short interactions (perhaps a few minutes at a time) for millions of users. A car-based conversational system, however, does not follow this model. Instead, longer interactions for a very few users, perhaps ten over the life of the car, are the norm. The need is for a personal assistant behind the dashboard that is observant, intelligent, proactive (especially when it comes to the driver's safety) and is capable of learning about the driver. Extant dialog manager-based systems can capture some of these features, but lack the fundamental structure to display all of them. To address this, the CRM architecture supports the easy integration of new services as well as rapid changes in the system's behavior, which allows for scaling up from a reflexive dialog manager to a more "butler-like" DM.

The CRM implementation shares many features of an operating system, e.g., scheduling, loading services, resource allocation, interprocess communication, etc. Services such as applications, speech engines, platform specific device drivers are attached to the CRM's *Engine Manager* component (see fig. 1) by simply declaring their name and parameters in a configuration file. This feature allows the CRM to support very rapid prototyping of VUI software by allowing engines providing these services to be attached and initialized at run-time.

The CRM allocates and schedules resources to avoid conflicts between services, according to policies written as a rule-base for another CRM component, the *Decision Control Process*. The CRM supports services such as reflexive VUI implementation, template-based natural language generation, attribute-value pair management, and User Interface description data management. Next generation systems can be built on the CRM base by supporting the CRM attachment protocol and extending the rule-base to describe the management requirements of the new services. At its core, the CRM consists of a device interface protocol, configuration data such as vocabularies and grammars, a rule base interpreter and a context history manager. Engines for specific functionality (e.g., DM's, navigation applications, etc.) can be added or removed at runtime. Modifying the vocabularies, grammars, prompts and Dialog Control Process rule base change the systems "personality." The context history vector stack can be analyzed to detect dialog breakdown and to facilitate limited learning.

Personality

Computer user interfaces, even simple command-line interfaces, are unconsciously perceived by users as expressing "personality." This phenomenon is unavoidable so it must be taken into account, and exploited if possible (Nass & Lee 2000). Personality is expressed in a conversational interface through several elements, including the vocabulary the system understands, the vocabulary and phrasing used by the system to express itself, and the approach of the system to an activity. For example, in response to a user's "It's hot in here", a passive system may perform no action. A more facilitating "butler" style system might ask, "Shall I make it cooler?" A pro-active "assistant" system might reply, "Okay, I turned up the air conditioning." Experimental results have shown that people prefer dealing with a user interface that has a personality which is similar to their own (Moon & Nass 1996; Nass & Lee 2000).

A haphazard user-interface without a clear personality makes it difficult for the user to accurately predict the system's capabilities and behavior. Inconsistency between the functionality that the user perceives in the system, and the actual proffered functionality can lead to dialog breakdown at a higher level than recognition error. With the choice of several "personalities," any user should be able to find a good fit for their desired interaction style.

Distinct personalities can be implemented in the PCA system by changing the reply grammars (for example from terse to verbose, amiable to business-like, etc.), and the degree to which the system chooses to present information to the user. The tone of voice in which the reply is delivered is also indicative of personality to the extent that TTS systems will permit word-level manipulation of emphasis and inflection. Finally, automatically detecting the best personality style for facilitating user interaction may be possible. Implementing this is difficult because it requires measuring the user's satisfaction with the system, as well as the degree of trust. This task is difficult for humans since it requires gauging emotional responses, body language, etc., all of which are skills acquired over years of cognitive development.

Dialog Breakdown Detection

Dialog Breakdown occurs when the behavior of a conversational system fails to correspond to the expectations of the user. In some portion of these events the user is able to recover. In others, mechanisms must be present in the conversational system to deal with the breakdown. Since breakdown is part of the conscious experience of the user, its existence can only be inferred by the conversational system. Reliably inferring dialog breakdown can be difficult prior to its becoming catastrophic. Obtaining information from many parts of conversational system can improve the reliability of such inferences.

The main sources of information, the Automatic Speech Recognition (ASR) engine, Task Manager, application and the content of the user's utterances are discussed below. In general, preventing dialog breakdown is less demanding to the user than dialog repair, but with the current technological limitations, misrecognition is unavoidable.

Speech Recognition failure Speech Recognition failures may be characterized by kind and density. The ASR may fail to find any match to the utterance, it may find matches of low confidence, or, it may find an erroneous match (which does not correspond to the user's utterance). The density of failures should be treated as data when monitoring the health of the dialog. Even a short consecutive sequence of failures may be sufficiently frustrating to the users confidence in the conversational mode of control.

Failure to recognize an utterance sometimes occurs even in human-to-human conversation. Outside of conditions that warrant urgent rapid actions, a limited number and density of recognition errors is tolerable. In human-machine communication, any failure to recognize the user should be acknowledged by the system to avoid having the user say the same thing more loudly (the tourist in a foreign land maneuver) which in turn may adversely affect recognition.

Recognition with insufficient confidence also occurs in human-to-human conversation. The human strategy for repairing the dialog is applicable. The conversational system on receiving a low confidence recognition result for, say, a phrase containing the word "X," can repeat the phrase with an emphasis placed on the word "X." User uttered affirmation phrases should allow the system to continue.

Erroneous recognition is dangerous to the value of a conversational interface even when the density of errors is low, since some acts cannot be undone, and acts that can be undone still place a burden on the user to coordinate repealing the act. The least that can be done is to carefully screen vocabularies to remove confusable commands, and to avoid executing the irreversible without a confirmation.

The CRM is built so that the density of recognition errors of all kinds is tracked and this density is used as a trigger for dialog repair processes. Systemic problems such as microphone button mismanagement, out of vocabulary speech, low speech volume, or excessive environmental noise have to be dealt with before the input speech recognition process can produce useful data. Where reliable means for distinguishing systemic problems exists, specific educational responses can be given to the user. This narrowing of instruction is very desirable, as is formatting the instruction to assign blame to the machine rather than the user.

Application detection of breakdown Assuming a system in which task expertise is embodied in either a Task or an intelligent application, an inappropriate command stream can be used to infer a dialog breakdown. Inappropriate command streams include commands that are not in the application or task repertoire or commands that are already being executed or are inappropriate at the present stage of execution. Such events should cause the conversational system to increment whatever values it stores in order to track the current dialog integrity.

As with speech recognition errors, the density of inappropriate commands is significant and should be tracked. Failure to understand a user's possibly novel utterance as a well-formed command reduces the users confidence in the system. In a conversational system that includes an acoustic add-word capability, a dialog can be initiated with the user to elicit both the novel and the standard utterance so that the novel utterance can be "understood" in the future. It is likely that the users behavior will be shaped by successful transactions with the system, but in absence of any learning on the part of the system some users will never be able to elicit some system behavior. Such holes in the system behavior may appear at critical points, rendering the system useless to some people.

User behavior, breakdown specific vocabulary In conversational systems, provision must be made for either the human or the machine to take initiative by recognizing the dialog breakdown state. Users can be expected to detect dialog breakdowns and to react with stereotyped verbal behavior. A breakdown detection vocabulary with good coverage of the scope of such behavior can be implemented.

In addition to utterances in the form "(expletive) computer" or "No, I said (valid command)" or "You dont understand," dialog breakdown may be indicated by frequent requests to the system for help or for repetition of its last prompt. Help is dialog repair before the fact. The user has recognized dangerous ground and is trying to avoid it. Help systems should be capable of increasing or decreasing the information they provide based on the history of their use.

The ultimate sign of dialog breakdown is the abandonment of the speech interface. This is detectable through the use of a timer that starts when the application system is turned on. Times and counts can be stored in non-volatile memory and tested against threshold values. Dialog could be re-initiated by one-sided verbal behavior on the part of the conversational system. Examples of such behavior might include status reporting and suggestions about useful command vocabulary.

Dialog Repair The best model for dialog repair behavior is naturally found in human-to-human transactions. Many methods used by humans are presently unavailable for simulation by conversational machines. These methods include, but are not limited to, deictic reference (physical pointing at an element of the physical context), metaphor, simile and other means involving higher order functions. However, human-like dialog repair can be simulated in general using simpler techniques.

When repairing the dialog, the system can revert to simplified vocabularies and behavior until the conversation is working again. If the repair behavior is stereotyped then the user will be able to detect that the system has been confused and react to it in a socially appropriate manner. Stereotyped behavior may include means for:

- Stepwise confirmation of commands and other means to guide use to correct utterances.
- Detecting and suggesting repair of acoustic environment.
- Repairing the system's vocabulary assumptions.
- Repairing the user's assumptions of system capabilities

In each case it is important to pay attention in the design to the social needs of the user. Changed behavior must be requested, not demanded. Step by step guidance can be provided if the user affirms their willingness to cooperate. Ultimately, dialog-repair should be taken as a last resort as mis-recognition of user intent is the main source of user frustration.

Limited Learning and Autonomy

One of the most significant difference between the PCA and previous dialog-related projects at IBM is that the transactions between the user and the system will be protracted rather than command and control or task completion. In such an environment a major benefit to the effectiveness of the interaction would be a system that actively learns user preferences and context to some degree.

The current PCA system is already capable of limited forms of learning. For example, the dialog manager component is capable of learning vocabulary mappings (e.g., "Yorktown" means "IBM TJ Watson Research Center") from the user. It is critical that this type of learning also takes into account the current context features. For example, the user might want the system to remember the mnemonic "MacDonalds" to represent the MacDonald family; however in the restaurant context, this utterance would take on a different meaning.

The PCA system is also capable of acting autonomously, in response to changes in the context vector in response to, say, messages from component engines, timers, etc. Depending on the chosen personality, the user can allow or almost fully suppress this autonomy. However, in some cases, e.g., if there is engine trouble, autonomy should not be suppressed for user safety. Outside of these cases, changing road and weather conditions, incoming phone calls, etc. can all be suppressed based on the system's personality.

Future Work

The long-term goals of the PCA project involve adding more autonomic, and learning/inference capabilities to the system. Two major capabilities are automatic adjustment of the dialog interface to suit the user's apparent skill level and task-recognition through user-modeling (e.g., recognizing the driver's commute route).

User-Skill Based Dialog Adaptation

Another capability emergent in the PCA system is user skill measurement, a weak form of user-modeling that attempts to gauge from past and current interactions the knowledge that the user has of the system. This information can be used to suggest to the user other functionality he or she may find helpful, or even to dynamically alter the dialog complexity in order to be more accommodating.

Unfortunately, skill cannot be measured directly in real time. The user's domain model, planning and execution are hidden, leaving only the action end of execution to be observed. Even observing the user's actions, the evaluation of the user's skill is confounded by the evaluator's implicit value judgments conflicting with user's preferred interaction style. Further, several non-observables factors such as luck, user's current state of mind, distinguishing successful goal completion from user acceptance of what is provided, abandonment from failure, reading from the manual from unassisted use, etc., make such evaluations unreliable. Even if these factors could be decoupled from measuring pure skill, such a measurement would still require the system to build and maintain a model of the users own cognitive model of the system.

If skill cannot be measured directly, then perhaps it can be approximated from interactions by studying the *dialog health*. Dialog health is a running measure of how well the component services (ASR, DM, CRM, Applications) evaluate their working state. If the ASR, for example, returns consistently high confidence scores then, even if the recognition is an error, then that component is "in good health." If it is mistaken, other components will soon report that their "health" is poor. Dialog health is calculated as some function of the number of ASR mis-recognitions, microphone button timing errors, dialog non sequiturs or topic switches, multiple uncompleted simultaneous tasks, multiple utterances of "repeat that," etc, and uses of harsh or abrasive language. Measuring these features will give some indication of whether or not the system is being used efficiently, and thus indirectly the skill of the user. Ultimately however, as the speech interface becomes more advanced, and more interaction styles and utterances are supported, the need for measuring user skill disappears, since any interaction style will find support, and the user will be free to frame his or her desires without "learning" the systems interface.

Long Term Learning and User Modeling

An actual in-car system will only interact with at most a handful of users at any one time. After a user has operated this tool for some initial duration, interaction will remain consistent over relatively long periods of time. This creates an opportunity to perform accurate user-modeling.

User modeling can be implemented as an engine that continually scans the CRMs context vector history, similar to the dialog breakdown detection engine, but with a longer temporal window. In this way user modeling can be performed purely on an abstract level, independently of the installed applications. When the user modeling engine recognizes a unique pattern or co-occurrence of states, the context vector can be translated back into application invocations, attribute-value pairs can be generated and fed to the dialog manager for translation into a request to the user.

Using this method, anything from simple macros (e.g., the driver always asks for the current traffic situation after starting the car) to more complex usage patterns (e.g., the driver commutes to work on a certain highway on weekdays) can be learned. The first example requires simply detecting causality in context vector states, in other words state vector A is always, or often followed by state vector B. In this case the system could ask, "Would you like me to report the traffic after you start the car?"[1] The second example can be realized in a similar manner using the context history vector, albeit looking for patterns on a different scale.

It may also be important for the system to pay attention to new or unique contexts. These may be indicative of a new driver, a new route to work, a vacation, etc. If the new context state persists, the system could prompt the user with, e.g., "Are you taking a trip?" Outliers in driver behavior are important because they may indicate a situation where the driver is need of more information than usual (i.e., the driver does not need the navigation system to drive to work, but may need it on an unknown route).

Other, more direct forms of learning can be made available. If for example the user wishes to define a new command, it can be constructed by simply compounding commands and providing a unique name.

Conclusions

A major concern when adding rich, interactive functionality in-car is that the driver may fail to pay adequate attention to the task of driving. Current systems use speech systems to partially address this problem, however as functionality increases, a simple command and control dialog system becomes increasingly irritating to the driver. The PCA builds on simpler dialog management systems to allow more human-like interaction, which address some psychological factors that cause user frustration.

[1]A more proactive personality might begin to report the traffic automatically without prompting the driver.

Acknowledgments

The authors would like to acknowledge the encouragement and assistance of Dr. Roberto Sicconi, Manager, Mobile Speech Solutions, IBM Research. We thank our colleagues in the Mobile Speech Solutions group at IBM for providing us with invaluable comments during our interactions.

References

Comerford, L.; Frank, D.; Gopalakrishnan, P.; Gopinath, R.; and Sedivy, J. 2001. The ibm personal speech assistant. In *Proc. of IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

Curry, D. G. 2001. In-vehicle cell phones: Fatal distraction - yes or no? In *Proc. of the Human Factors and Ergonomics Society 42nd Annual Meeting*, 562–566.

Gellantly, A. W., and Dingus, T. A. 1998. Speech recognition and automotive applications: Using speech to perform in-vehicle tasks. In *Proc. of the Human Factors and Ergonomics Society 42nd Annual Meeting*, 1247–1251.

Greenberg, J.; Tijerina, L.; Curry, R.; Artz, B.; Cathey, L.; Grant, P.; Kochhar, D.; Koxak, K.; and Blommer, M. 2003. Evaluation of driver distraction using an event detection paradigm. In *Proc. of the Transportation Research Board Annual Meeting*.

Lee, J.; Caven, B.; Haake, S.; and Brown, T. 2001. Speech-based interaction with in-vehicle computers: The effect of speech-based e-mail on driver's attention to the roadway. *Human Factors* 43:631–640.

McCallum, M.; Campbell, J.; Richman, J.; and Brown, J. 2004. Speech recognition and in-vehicle telematics devices: Potential reductions in driver distraction. *International Journal of Speech Technology* 7:25–33.

Moon, Y., and Nass, C. 1996. Adaptive agents and personality change: Complementarity versus similarity as forms of adaptation. In *Proc. of SIGCHI*, 287–288. ACM Press.

Nass, C., and Lee, K. M. 2000. Does computer-generated speech manifest personality? an experimental test of similarity-attraction. In *Proc. of SIGCHI*, 329–336. ACM Press.

Redelmeier, D. A., and Tibshirani, R. J. 1997. Association between cellular-telephone calls and motor vehicle collisions. *New England Journal of Medicine* 336:453–458.

Reeves, B., and Nass, C. 1996. *The Media Equation: How People Treat Computers, Televisions, and New Media Like Real People*. Cambridge University Press.

Shechtman, N., and Horowitz, L. M. 2003. Media inequality in conversation: How people behave differently when interacting with computers and people. In *Proc. of SIGCHI*, 281–288. ACM Press.

Violanti, J. M. 1998. Cellular phones and fatal traffic collisions. *Accident Analysis and Prevention* 30:519–524.

Wickens, C. D. 1984. Processing resources in attention. In Parasuraman, R., and Davies, D. R., eds., *Varieties of Attention*. London: Academic Press.