Dialogue-based human-computer interfaces and active language understanding

Will Fitzgerald, Kalamazoo College and I/NET, Inc. will.fitzgerald@kzoo.edu

R. James Firby, I/NET, Inc. firby@inetmi.com

Recent developments in speech, network and embedded-computer technologies indicate that human-computer interfaces that use speech as one or the main mode of interaction will become increasingly prevalent. Such interfaces must move beyond simple voice commands to support a dialogue-based interface if they are to provide for common requirements such as description resolution, perceptual anchoring, and deixis. To support human-computer dialogue effectively, architectures must support active language understanding; that is, they must support the close integration of dialogue planning and execution with general task planning and execution.

Key words: human-computer interface, dialogue planning and execution, active language understanding

Will Fitzgerald is an associate professor of computer science at Kalamazoo College and an adjunct research scientist at I/NET, Inc.

R. James Firby is the chief scientist of I/NET, Inc.

Dialogue-based human-computer interfaces and active language understanding

Will Fitzgerald, R. James Firby

The need for dialogue-based human-computer interfaces

Although the ability for machines to interact with humans via language has been an aspiration since before the computer era, recently, some human-computer interaction researchers, such as Shneiderman (2000), have called into question the desirability of such a thing. And it is true that visualization tools and graphical human-computer interfaces have come to dominate the ways computer users interact with computers, especially in the ways people interact with a "personal computer," that is, one person interacting with a desktop or laptop computer with a high-resolution display.

Still, several trends suggest that speech-based interfaces will play an increasingly important rôle in human-computer interaction. First, there has been a return to statistical models of speech recognition, and this has proved to be a successful strategy for building more reliable continuous speaker-independent speech recognition. Second, the increasing power and miniaturization of computing power and memory has enabled faster and more ubiquitous availability of speech recognition technologies. Third, the same trend towards smaller, faster, cheaper computers has meant that there are many more embedded computers requiring interfaces. Because these devices are built to be embedded and because of the trend towards miniaturization, large, high-resolution displays are not as available. Fourth, the ubiquity of cell phone technology means many users of high-technology are carrying increasingly powerful microphones. (Of course, cell phones themselves are small computers and have small displays and so are themselves likely targets for speech interfaces). Fifth, the advent of cheap wireless networking and adoption of standard wireless protocols means that many more devices can interact, and thus even small devices can participate in the world-wide network of computers. This implies, for example, that a device could call on another device for dialog interface services–or to be controlled by such a device.

What we are likely to see is increasing use of multimodal human-computer interfaces, in which speech-based interfaces naturally and seamlessly interact with interfaces based on human visual, tactile and other senses. The question is, what is required to create effective speech-based interfaces? Clearly, these interfaces require excellent microphone technology, "speech-to-text" technology, and "text-to-speech" technology. In addition, effective speech-based interfaces will also require the ability for humans and computers to engage in *dialogue*: that is, people and computers will need to interact via spoken language (and other modalities) to make requests of one another, to question one another, to inform one another, and so forth. Equally importantly, people and computers will need to interact to clarify, deny, cast doubt on, etc., the requests, questions, informings, and so forth, being made.

In other words, much of the richness available in human-human interaction will be important for human-computer interaction. Because human dialogue is typically fluent and unreflexive, once computers engage to any extent in dialogue, people start by assuming computers have the same fluency in dialogue they themselves have. Computers will not have such fluency any time soon (if ever), and thus the resources for handling conversational breakdowns available to humans (for example, asking for clarification) will need to be available to machines as well. Further, it is part of the power of human dialogue that deixis, ellipsis, anaphora, etc., are available, and to deprive the human-computer interface of these capabilities is to limit its effectiveness and power. Advances in speech-based interfaces will require advances in dialogue-based interfaces.

Architectures for dialogue and action

Given the goal of building "engines" for handling human-computer dialogue, what kind of architecture would best provide a blueprint for building such engines? First, it has long been recognized that engaging in dialogue is a planful activity (Schank and Abelson, 1977; Cohen and Perrault, 1979). That is, dialogue is an activity which involves the interlocutors' goals and plans to achieve those goals. If I want some trash picked up, for example, I might pick it up myself. On the other hand, I just might point to the trash, and expect someone else (a robot or a servant, say) to infer that it should be picked up. Or I might engage in a dialogue plan, and request that someone (a robot or a servant, say) pick up the trash. The point is simply this: engaging in dialogue is engaging in a particular kind of task, intertwined with all the other tasks, goals, plans, resources, etc., with which an agent is engaged.

Thus, dialogue shares many of the same requirements with other kinds of task execution. For example:

- Agents can have multiple goals active simultaneously,
- Agents can be attempting to achieve these goals simultaneously,
- The world in which agents act is often unpredictable and often changing,
- Tasks and goals can be described hierarchically,
- Resources available for achieving goals vary,
- Goals can have differing priorities, which may vary dynamically,
- Agents will sometimes fail to achieve their goals,
- Multiple agents will be attempting to achieve their goals simultaneously, perhaps collaboratively,

and so on.

It is true that engaging in natural language dialogue has different qualities from non-dialogue tasks. For example, natural language is rife with vague descriptions whose meanings must be negotiated; dialogue almost immediately requires one to address issues of other agents' beliefs, goals, and intentions. However, it is equally true that many non-dialogue tasks share these same qualities. For example, we have argued that descriptions have value for tasks beyond natural language dialogue (Fitzgerald and Firby, 1996). In the end, dialogue planning and execution is task planning and execution.

Task execution is best done actively

When the environment in which tasks are carried out is relatively simple and static–either due to the nature of the environment or because it has been engineered to be simple and static–traditional approaches to planning and plan execution may suffice. That is, a planner may be able to project from a set of axioms and rules to a precisely defined and ordered series of tasks for an agent to carry out. For example, a person writing a computer program in an imperative style (in programming languages such as C or Java) is, essentially, asserting a set of axioms from which a compiler deduces the precise machine or byte code instructions to carry out, and in what order. The "world" of machine instructions, being relatively simple and static, lends itself to this approach.

In other cases, the environment in which tasks are carried out is often complex and dynamic, however. By "complex," we mean that the future state of the environment is often unknowable, even given full knowledge of the current state (see, for example, Wolfram 2002). By "dynamic," we mean that the environment is likely to change without our knowledge, which is just another way of saying that we cannot have full knowledge of the state of an environment.

For example, if an agent (human or robot) plans to fetch a glass of water from another room, the agent cannot plan, down to the level of its sensing and acting primitives, an exact series of tasks which will guarantee its success in fetching the water. Yet, humans succeed in fetching water every day, even if robots cannot yet do so. Why is this so?

Humans have the ability to act and react to the environment even as they carry out tasks. The ability to *actively engage the environment* is a crucial ability for successful action. This suggests several features of a successful task execution system:

1. Plans can be *sketchy*, that is, the tasks to be carried out must be describable at appropriate levels of abstraction. For example, a task in a plan to fly to New York is "get on an airplane going to New York," with the parameters (which airplane) and specific details (how to enplane at a particular airport) abstracted away.

2. Plans can have *multiple ways to succeed*; and, having multiple methods to achieve success, must carry the *contexts under which each method is appropriate*. For example, a plan to open a door might have different behaviors which depend on the type of door handle.

3. Plans must carry their *success and failure conditions* which are checked while tasks are being carried out ("run-time conditions"). Because the world is complex and dynamic, success and failure must be determined at run-time, not just projected at compile time.

4. Plans can provide for *breakdown*, that is what to do when failure occurs. For example, should an agent fail to get on an airplane going to New York , the agent can have methods for coping with this (check with a travel agent for another flight, for example).

In any case, any architecture for task execution in complex, dynamic environments must allow an agent to be *active* in the sense required: the environment itself must be factored into task execution.

Dialogue-based human-computer interfaces require active architectures

If dialogue is a kind of task, and task execution requires a task execution architecture that is active, it follows that the successful execution of dialogue also requires a task execution architecture. The multi-agent nature of dialogue makes this imperative, as an agent's interlocutor is not under the agent's control and the interlocutor's beliefs, goals, and actions are especially opaque; also, the agents are attempting to achieve their goals simultaneously, with only loose synchrony. In addition, an agent engages in dialogue to help achieve other goals, not (typically) just to engage in dialogue. Hence, dialogue tasks must take place in the context of other tasks (both linguistic and non-linguistic), sharing state, as well as "cognitive" and physical resources.

For example, consider a dialogue-based human computer interface to a computer system that can control temperature set points inside of, say, an automobile. If the driver says, "I'm too hot," the system should interpret this as statement of the driver's goal to be cooler. The system can then engage in the (non-linguistic) tasks which will achieve this goal, by lowering the temperature set point, and perhaps engage in the (linguistic) task of informing the driver of the new set point. If the driver then says, "Too much!" the system should interpret this as a failure to adopt the appropriate goal for setting the temperature, and should then attempt to increase the set point somewhere between the old set point and the next set point, given the context of the conversation (a linguistic context) as well as the new set point (a non-linguistic context). In another context, "Too much!" could be interpreted differently; for example, it might mean that the temperature set point temperature should be decreased rather than increased. Or, in the context of a request to increase the radio volume, it should be interpreted as a failure in properly setting the radio volume. And so on.

That dialogue requires active task execution entails the requirement for what we call *active language understanding*. Active language understanding is a view of pragmatic interpretation which emphasizes the close relationship that language and dialogue have to other kinds of activity, including the planful nature of language (i.e., that it is undertaken to achieve the goals of its interlocutors) and the need to engage actively in the linguistic and non-linguistic context in order to achieve understanding.

Examples

To illustrate active language understanding in a dialogue-based human computer interface, we provide three examples: description resolution, perceptual anchoring of symbols, and deixis. We use for the examples which follow the AERCam Sprint (NASA 2002), equipped with a voice interface. AERCam Sprint, a spheroid free-flying robot containing two television cameras, an on-board navigation system, and 12 small nitrogen gas-powered thrusters, is designed to allow remote inspections of the exterior of International Space Station. The examples come from our work on building an AERCam Sprint test bed, which combined a simulation of the free flyer in a shuttle deployment along with a mixed-mode graphical and dialogue-based control station (Fitzgerald and Firby, 1998). Although we use AERCam Sprint as our example, the same general active language understanding issues often obtain with other systems.

Description resolution

Consider the following interaction between an astronaut and the AERCam Sprint control station. The shuttle has four elevons (a controller that combines the functions of an elevator and an aileron), two on the port wing, and two on the starboard wing. The two elevons closest to the shuttle body are the inboard elevons, and the two further away are the outboard elevons. The dialogue occurs in a relatively neutral environment: the astronaut and the free-flyer are both in the shuttle bay (i.e., equidistant from the inboard elevons) and they have not been discussing the elevons.

Astronaut: Go to the inboard elevon.

AERCam: There are two inboard elevons. Which one?

Astronaut: The port one.

AERCam: [Flies to the port, inboard elevon.]

Astronaut: Look at the outboard elevon.

AERCam: [Rotates to point its camera at the port, outboard elevon.]

In the first command, the description "inboard elevon" is ambiguous in the current neutral context: it could mean either of the inboard elevons. Thus, the AERCam Sprint control station asks the astronaut which was meant. The description "The port one" is used to further disambiguate to the unique port elevon. When the second command ("Look at the outboard elevon") is issued, AERCam Sprint has moved in close proximity to the port outboard elevon and moved far away from the starboard outboard elevon. This fact is enough to disambiguate the description "the outboard elevon" in the new context to the outboard elevon.

Within a complex, dynamic environment, it is impossible for an active language understanding system to be completely sure its heuristics for description resolution will produce "sound and complete" results. This has to do with the nature of the environment, not with active language understanding. Still, standard heuristics for description resolution exist. Among these are:

- *Focus of attention*. Objects often become the focus of attention due to conversational context.
- *Spatial proximity*. Knowing the physical proximity of an object is often important for disambiguation. "Track the elevon" is more likely to refer to a closer object than to a far away object; "fetch a halon drum" is more likely to refer to an object far away.
- *Sensed objects*. Knowing which objects an agent can sense is often relevant, as is knowing which objects other agents can sense. "Track the elevon" is more likely to

refer to an elevon that the astronaut making the request can see than to one the astronaut cannot see.

- *Containment and partonomic relationships*. Knowing whether an object is within the same container is often relevant. "Track the communications satellite" is more likely to refer to a satellite within the payload bay if AERCam is also in the bay. Knowing partonomic relations (which objects are part of other objects) is often relevant. "Track the tip" is more likely to refer to the tip of a particular object if that object is the current focus of attention.
- *Type of action being performed*. Knowing the type of action to be performed on the object is often important, perhaps for allowing or disallowing the use of other heuristics. Knowing that the action is "fetching" may mean considering objects further away, in other containers, etc.
- *Uniqueness of resolution*. Knowing that a description is uniquely satisfied is a powerful, but often expensive, heuristic. If the agent knows of only one blue halon drum in the world, it is often the case that this is the drum being referred to.
- *Tasks for further disambiguation*. An agent can also directly seek additional information to disambiguate descriptions. In the example above, the AERCam Sprint control station initiated a sub-dialogue to clarify which port elevon was intended.

These heuristics are not mutually exclusive. Objects one can sense tend to be nearby, for example.

Perceptual anchoring of symbols

Perceptual anchoring is "the process of creating and maintaining the correspondence between symbols and percepts that refer to the same physical objects" (Coradeschi and Saffiotti, 2001). AERCam Sprint, for example, given the goal of tracking a specific astronaut (that is, to maintain the focus one of its cameras on the astronaut) must maintain an active correspondence between its internal representation of the astronaut and its percepts as the astronaut moves along, as well as does the device itself.

Perceptual anchoring is, in many respects, the same as description resolution. For example, "Track *the tall astronaut*" may resolve the description "tall astronaut" to an internal symbol `astronaut-2`, in the ways described in the previous section. Similarly, perceptual anchoring builds and maintains correspondences between the symbol `astronaut-2` and percepts which identify her or him (say, for example, the color of his/her spacesuit or other distinguishing marks).

At the extremes, linguistic descriptions are a part of a relatively static social sign system, while perceptual anchors can be defined dynamically, idiosyncratically and non-symbolically (see for example, the explanation of iconic and geometric models in Pirjanian and Christensen, 1995). On the other hand, novel linguistic descriptions can be dynamically created ("the port outboard elevon") or even invented as nonce expressions ("let's call it 'elevon 1'"). Further, perceptual anchors can take on social symbolic status (the captain's insignia, or an agreement that astronauts will always where the same color). Additionally, the kinds of heuristics described previously which can be used by an "active language understanding system" can also be employed by an active perceptual anchoring system" to acquire and track objects, using heuristics such as spatial proximity, knowledge about containment, and partonomic relationships.

Deixis

All interpretation of deixis requires knowledge of the context in which the deixis is used, and the context is often, if not usually, the physical environment. "Go to *that* elevon," requires interpretation with respect to, among other things, the elevon to which the speaker is pointing, how the speaker is pointing to an elevon, and the visual fields of the speaker and hearer. Crucially, the hearer has to track along the path defined by the speaker's pointing. This tracking must require activity on the part of the hearer; it cannot be known *a priori*. Further, if the hearer has only the speaker and not the object being pointed to in the hearer's visual field, the hearer will typically have to move its visual system along the tracking path until it spots the object being pointed to.

Note that deixis can combine description resolution and perceptual anchoring. "Go to *that* inboard elevon" requires resolving the description of "inboard elevon" as "perceptual anchors" are used to scan for objects along the path for which the "inboard elevon" description makes sense.

These three examples–description resolution, perceptual anchoring, and deixis interpretation–illustrate the need for active language understanding: any system that implements these features of human language understanding will need the ability to sense and act in its world.

Are dialogue-based human-computer interfaces feasible?

We have demonstrated several instances in which a system which adequately understands language and dialogue must be able to be an "active language understanding" system; that is, to take action with respect to its environment in order to understand and generate language in context.

Because we already know that humans use dialogue for interaction, in the limit, building a dialogue-based human computer interface is of the same order of difficulty (quantitatively as well as qualitatively) as building an agent that is as intelligent as a human,. Thus, it may appear impossible to build feasible dialogue-based human-computer interfaces. On the other hand, as we have argued, a voice-based interface to be will have to have aspects of active dialogue to be effective, both on account of the human proclivity to ascribe human capabilities to computers, and because they are frustrated when their expectations are not met; but also and mainly because of the inherent advantages of dialogue (the ability to use general descriptions, engage in clarifying dialogue, etc.).

One test of feasibility is a practical one–can useful systems be built that incorporate dialogue-based human-computer interfaces? Over the past several years, we have in fact built such systems. For example, we have built an interface for the AERCam Sprint Testbed

(Fitzgerald and Firby, 1998). Successive generations of this interface have been incorporated into a tool for system monitoring and for control of devices in automobiles (I/NET, 2002). In doing so, we have built up a significant package of dialogue plans for human-computer interaction, description resolution, and so on. In building these systems, there are a number of lessons that we have learned along the way. For one thing, it must be admitted that, although building such systems is feasible, it is often daunting in scope. One strategy we are currently exploring is the use of "conversational interaction domains" (Firby, 2002) that define human-computer dialogues for a wide variety of systems within a specific domain. For example, consider the "device control domain," which describes dialogues about device control: turning devices on or off, creating a set point (such as a thermostat setting), and so on. Dialogues about device control are a common requirement for human-computer interaction, and thus, from an engineering point of view, building dialogue-based human-computer interfaces becomes much easier if a system designer can view his or her task as one of configuration (for example, describing the devices to be controlled in a particular domain) rather than programming the interface *de novo*.

Conclusion

Dialogue planning and execution are not essentially different from other planning and task execution domains. On the one hand, it is true that building dialogue systems immediately bring to the fore such issues as agents' beliefs, desires, and intentions as well as problems of ambiguity/vagueness. On the other hand, however, these issues are important for other domains as well, such as story understanding, building educational systems, and so forth. Further, the same issues that arise from executing tasks in a dynamic, unpredictable world also arise in dialogue planning and execution; for instance, ambiguity and vagueness arise in other sensor modalities, too. For example, an agent may have to change its point of view to eliminate ambiguity in a visual scene. Our experience in implementing systems which handle issues such as description resolution, perceptual anchoring, and deixis further underscores the similarities between dialogue execution and general task execution. This strongly argues for an architecture which integrates active language understanding with active task execution in order to create effective dialogue-based human-computer interfaces.

References

Cohen, Phillip R. & C. R. Perrault (1979). Elements of a plan-based theory of speech acts, *Cognitive Science*, 3(3):177-212.

Coradeschi, Silvia & Alessandro Saffiotti (2001). Perceptual anchoring of symbols for action. *Proceedings of the 17th IJCAI Conference*, 407-412. Seattle, WA.

Firby, R. James. (1995). Adaptive execution in complex dynamic domains, Yale University Technical Report YALEU/CSD/RR #672.

Firby, R. James. (2002). Conversational interface domains. URL: http://sbir.gsfc.nasa.gov/SBIR/abstracts/01/sbir_phase2/013653_abstract.html.

Fitzgerald, Will. & R. James Firby. (1998). The Dynamic Predictive Memory Architecture: Integrating language with task execution. *Proceedings of IEEE Symposia on Intelligence and Systems (SIS '98)*, May 21-23, Washington, DC.

Fitzgerald, Will & R. James Firby (1996). Item descriptions add value to plans. *Proceedings of The Workshop on Plan Execution: Problems and Issues of the AAAI Fall Symposium*, San Mateo, CA: Morgan Kaufmann.

I/NET (2002). The conversational interface. URL: http://www.inetmi.com/ci/.

NASA (2002). AERCam Sprint. URL: http://spaceflight.nasa.gov/station/assembly/sprint/.

Pirjanian, Paolo & Henrik I. Christensen (1995), Hierarchical control for navigation using heterogeneous models, in H. Bunke, T. Kanade & H. Noltemeier (Eds.) *Modeling and planning for sensor based intelligent robot systems,* (pp 344–361). Machine Perception and Artificial Intelligence series. Singapore: World Scientific.

Schank, Roger and Robert Abelson. (1977). *Scripts, plans, goals and understanding*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Shneiderman, Ben. (2000). The limits of speech recognition. *Communications of the ACM,* 43 (9), 63–65.

Wolfram, Stephen (2002). *A new kind of science*. Champaign, IL: Wolfram Media, Inc.