Goal-Oriented Dynamic-Initiative Dialog and Action Management

Cody Canning

May 14, 2013

1 Introduction

Natural language dialogues with robots are an important step toward more natural human-robot interaction (HRI). More and more often humans and robots are coordinating to accomplish complex, sometimes dangerous or high-risk, tasks. By interacting in natural language with robots during these tasks we remove the overhead of learning a specialized language and promote natural attitudes and behaviors that are more sustainable. As robots make their way into our homes and workplaces it is imperative to ensure that the robots are prepared to exist as entities situated in physical environments. Moreover, it becomes possible (and indeed necessary) to utilize perception and action in and of the world as tools for improving the spoken natural language abilities of robots.

Mixed-initiative human-robot team tasks require coordination and cooperation between the members of the team. Some of the features necessary to do so include (1) a shared pool of knowledge about the task environment, the task goals, task-related objects, etc., (2) smooth, actionable information transfer between members (clear, understandable dialog), and (3) individual responsibility and initiative with regard to the task. This paper presents an approach toward developing the dialog management (and to some extent the natural language generation and action management) component of a sophisticated, situated robot architecture, which is designed for and evaluated in a specific human-robot team task scenario.

This architecture bootstraps its own perceptual content so as to expand primarily upon (3), above. Using perceptual information to drive the robot's initiative creates a more natural dialog because the robot has inherent resourcefulness and "enthusiasm", in a way, for completing the task. This energy manifests in the content of both the robot's behavior and speech. Crucially, this architecture improves task efficiency as well as quality of interaction. A robot that takes the initiative to acquire and communicate goal-related information via perception and action increases the quantity of task-relevant dialog content and drives the task toward completion.

The following section describes the scenario in which this situated natural language understanding architecture is participating. Next is a description of the architecture, detailing the functionality at the algorithmic level of each component of the system. Following that is an evaluation of the

component and an analysis of its advantages and disadvantages. The last section is the conclusion which suggests improvements and future work.

2 Human-robot Team Task

The test bed for this novel architecture is a human-robot remote team task. The task takes inspiration from the paradigm described in the Indiana cooperative remote search task (CReST) corpus [1]. In a CReST task pairs of spatially separated humans communicated via audio links while accomplishing several mission objectives concerning target objects ("colored boxes"). These objects were placed throughout an indoor environment which consisted of office rooms and a hallway. One individual was the *director* and coordinated the task from a remote location while the other was the *searcher* and navigated through the office space.

The scenario being considered puts the robot into the role of the searcher in a stripped-down version of the CReST. The dialog takes place between a human director and the robot which exists in a simulated, two-dimensional environment consisting of rooms and a hallway. In order to function as a capable member of the team in this scenario the robot must be able to do at least the following:

- Follow the director's directions by going into specified rooms and perceiving objects by shape and color.
- Manage the dialog by responding effectively and quickly to task and environment-related yes-no questions, wh-questions, declarative statements, and interrupts.

The main objective of the task is to locate a blue box in the office environment. However, the director may be informed of new goals throughout the task. Since the task is mixed-initiative – that is, both team-members can "drive" the task forward – the robot is permitted to act and speak without an explicit command to do so.

There are two main strategies for engineering dialog managers that can effectively participate in this type of scenario:

Knowledge-based dialog management. The earliest approach to dialog management was knowledge-based [3]. This approach is best suited to specialized domains where the system can dictate the progression of the dialog. Finite-state automata were the dominant algorithm used, and the rules were generally handcrafted – that is to say, engineers explicitly programmed the state transitions of the automata to work for the domain in which the DM was being used – based on experiments with real users [2]. These types of systems were inflexible and it was difficult to cover the appropriate breadth of interaction. More recent expansions on these ideas include the agenda-based approach which segments tasks into smaller, simpler subtasks and separates the domain-dependent from the domain-independent aspects of the DM. However, this type of approach still requires difficult, time-consuming design by experts.

Data-driven dialog management. A more recent approach to dialog management is that of stochastic dialog modeling using reinforcement learning based on Markov decision processes (MDPs) or partially-observable MDPs (POMDPs) [3]. Instead of designing the algorithm around the data,

these approaches use statistical techniques to train dialog strategies on annotated databases. While annotating the data is still time-consuming, the algorithm remains domain independent. However, there are issues which arise from the data-drive approach. Chief among these is lack of control developers have over the optimized policy and the dialog progression.

Degree of initiative is a characteristic that differentiates potential solutions. *System-initiative* dialogues are dictated by the system at each step while *user-initiative* dialogues are guided by the human.

In some cases the robot might take virtually no initiative and exclusively respond to the interlocutor. Other solutions might have the robot frequently initiating new dialogues and/or moving around the environment without being prompted to do so by the human.

3 Goal-oriented Dynamic Initiative Dialog Operator

3.1 Architecture Components and Subcomponents

My strategy for developing a dialog manager for the aforementioned scenario is closer to the knowledge-based than the data-driven approach. It utilizes a knowledge-base that is flexible and domain-independent. But it requires policies to be defined for various aspects of the architecture that are specific to the task domain. The focus of this dialogue management system is on *dynamic initiative-taking*: goal-related information is propagated through the dialog to the interlocutor(s) as often as possible. In order to do so it is sometimes necessary for the robot to take actions in the environment so as to detect goal-related percepts. The details of this architecture are laid out below.

The dialog management robot architecture designed for the aforementioned scenario is called the Dynamic Initiative Dialog Operator (DIDO). DIDO is a component of the Agent Development Environment (ADE) [4] and leverages several of the other components therein (See Figure 1). Each of the components is part of (or complementary to) the situated natural language understanding pipeline and managed by the ADE Registry:

- `ADESimEnvironmentComponent`: generates the 2D office environment and its contents.
- `SimPioneerComponent`: differential-drive robot with door, box, and color-detection. Receives requests from `DIDOComponent` for percept information.
- `MotionComponent`: moves the robot around the environment. Receives driving directives from `DIDOComponent`.
- `SimSpeechRecognitionComponent`: provides a GUI for transmitting natural language commands to the robot via the keyboard.
- `CanningComponent`: parses utterances and packages them together with their semantics as "cans", which are then passed to `DIDOComponent`.

- `DIDOComponent`: manages the knowledge base, goals, actions, dialog and natural language generation.
- `MaryTTSComponent`: generates speech from the text supplied by `DIDOComponent`.

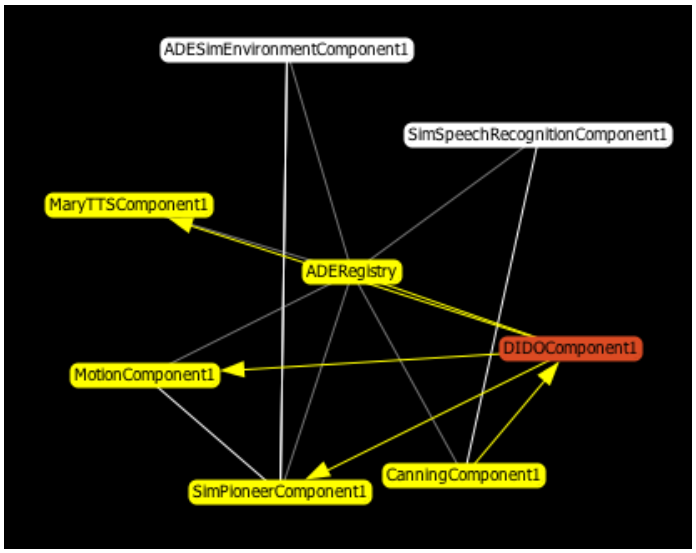

Figure 1: Components running within ADE. Arrows indicate inter-component dependencies for the `DIDOComponent`.

From the breakdown above it is clear that the `DIDOComponent` is responsible for a lot. The following subsections describe how it manages each facet of its functionality in the context of the search task.

3.1.1 Knowledge Base

The `DIDOcomponent` retains a knowledge base (KB) of entities and their properties. These entities arise principally through conversation with the interlocutor. Properties are represented as a list wherein each property has a corresponding list of values. Statements such as "there is a green box in the breakroom" add appropriate entities to the KB – in this case *green box* and *breakroom*. Sentence-specific entities are included in the packages received from the `CanningComponent`, along with the semantics of the utterance. There are mechanisms in the component for merging and disambiguating KB entities in the case that multiple entities with the same name and/or properties arise, the specifics of which are beyond the scope of this paper. In addition to entities learned about through dialog, the KB is regularly updated to include objects acquired through perception of the environment (e.g. boxes, doors, rooms, etc.)

3.1.2 Goals

Goals are represented in the `DIDOComponent` as a set and are regularly compared to entities contained in the KB. If the relevant properties of the goal match those of an entity in the KB then

a goal has been completed. For example, the robot might have a goal to find the *location* of the *blue box*. An algorithm that checks for goal completion will regularly scan the KB for a box entity with the blue *color* property and a non-null *location* property. If a scan were to return the *blue box* entity with *location* equal to *breakroom* then the robot would inform the interlocutor that a blue box is located in the breakroom and that a goal has been achieved. It would then remove this goal from the set of goals.

3.1.3 Dialog Management and Natural Language Generation

The `DIDOComponent` manages dialog reactively – it generates responses to the utterances of the interlocutor(s) – with the following key exceptions:

Goal-related percept reporting. If the robot perceives something in the environment it compares that percept to each of the goals in the robot's goal set. If the percept is *appropriately similar* to a goal-related entity, it informs the director. For example, if the robot has the goal to find the location of the blue box and it at any point perceives a green box, it (1) checks that *green box* has type box, which is the same type as goal-related entity *blue box*, and (2) generates an utterance that informs the director of the green box and its location. The policy for this specific scenario dictates that *type* is the relevant property for establishing appropriate similarity of entities.

Goal-oriented action execution. The robot will take the initiative and execute an action in order to answer a question. The criterion for doing so are described in the Action Generation section below. Once the decision to execute an action has been computed, the robot generates an utterance informing the director of its intentions in a natural way. For example, if the director asked "What is in the hallway?", the robot might respond, "Hold on, let me check", as it drives to the hallway.

Goal completion reporting. Whenever the robot accomplishes a goal, e.g. locating the blue box, it immediately generates an utterance to inform the director: "I have located the blue box in the breakroom". This is a highest-priority directive and the robot will interrupt the director if necessary.

3.1.4 Action Generation

Actions are generated based on both requests from the director and the robot's own initiative. In the former case the robot might be told, for example, to go to the breakroom. It then sends a request to the `MotionComponent` to drive to the breakroom. The robot can also take its own initiative to act in the environment. The decision to do so is based on current goals, what the director says, what is in the KB, and an *expected cost* function. The following pseudo-code illustrates this decision-making process:

for *goal* in *goals* **do**
  if `concernsGoal`(*semantics*, *goal*) **then**
    $action \leftarrow$ `generateAction`$(semantics, goal)$
    $ec \leftarrow$ `getExpectedCost`$(action)$
    if $ec <$ `costThreshold` **then**

```
            execAction(action)
            break
        end if
    end if
end for
```

The `generateAction` function takes as arguments (1) the semantics of the utterance received by the `DIDOComponent` and (2) a goal, and returns an action. The action is composed with the purpose of acting in order to derive the answer to a question. For example, if the first argument were the following predicate (location, blue box, breakoom) with the utterance type, QUESTIONYN ("yes-no question"), then the function would produce the action (drive, self, breakroom). Once the robot drives to the breakroom it will presumably be able to perceive any blue boxes and thus answer the director's question. Since the domain of this search-task scenario is quite limited, it was straightforward to implement this function as a policy table based on the function arguments.

Expected cost of an action is the criterion for deciding to execute an action (or not). The `DIDOComponent` computes an expected cost for each action in order to disallow actions that could have negative effects on the task. For example, the robot deciding of its own accord to investigate the contents of a room that is very far away could confuse (or anger) the director and waste valuable time. The expected cost is computed by the `getExpectedCost` function. This function takes one argument: the action in the form of a predicate, e.g. (drive, self, breakroom), and returns the expected cost by computing the distance to the destination (*breakroom*, in this case) from the robot's current location. The `costThreshold` variable represents the global threshold of action valuations (set to 10 meters by default).

This distance computation is a naive cost estimate, but for the search task it suffices. In other domains it will be necessary to adapt this function to account for the relevant considerations. Furthermore, it is imperative to ensure that the robot does not execute an action that jeopardizes the team's ability to complete goals in the future. Certainly, if the director asked, "Is there a blue box in the swimming pool?", the robot should not jump at the opportunity to drive into the swimming pool to investigate (assuming of course that the robot is not amphibious). This could be formalized as a "potential detriment" or "risk" variable.

3.2 Results from Sample Runs

Deploying the robot design detailed above into the CReST-like task scenario produced promising results. I have included comparisons of the `DIDO` architecture in specific scenarios to a system which does not take initiative in order to highlight the enhancements to both the efficacy and the flow (or *naturalness*) of the dialog.

Comparing dialog 1 to dialog 2 in Table 1 we see how the `DIDO` architecture produces a more natural dialog by checking the breakroom for the blue box on its own initiative[1]. A system not equipped to take initiative in this way would have to be explicitly told to go to the breakroom and

[1]This initiative-taking might also be described as acting on an implied request to check the breakroom for a blue box.

asked the same question twice. This comparison highlights the *goal-oriented action-execution* and *goal completion reporting* features of the `DIDOComponent`.

Table 1: In Dialog 1 the robot takes the initiative to act and sense, demonstrating *goal-oriented action-execution* and *goal completion reporting*.

Dialog 1	Action / Perception
D: "Our task is to locate the blue box."	
R: "Acknowledged."	
D: "The room in front of you is called the breakroom."	
R: "Understood."	
D: "Is there a blue box in the breakroom?"	
R: "Hold on, let me check."	(drive, self, breakroom)
D: "OK."	
R: "I have located a blue box in the breakroom."	percepts: blue box, green box

Dialog 2	Action / Perception
D: "Our task is to locate the blue box."	
R: "Acknowledged."	
D: "The room in front of you is called the breakroom."	
R: "Understood."	
D: "Is there a blue box in the breakroom?"	
R: "I do not know. I am not in the breakroom."	
D: "Go to the breakroom."	
R: "OK."	(drive, self, breakroom)
D: "Is there a blue box in the breakroom?"	
R: "Yes."	percepts: blue box, green box

We next compare dialog 1 to dialog 2 in Table 2 in order to illustrate *goal-oriented percept reporting*. The robot in dialog 1 running the `DIDO` architecture naturally informs the director of something *goal-like* in the breakroom: a *yellow* box. It turns out in this example that knowing about the yellow box allows the director to make an expedited decision to command the robot to go North. In dialog 2 the robot does not offer up goal-related perceptual information on its own, so the director has to query the robot for it. It might be the case that knowing about the yellow box does not aid in driving the task forward and only provides extra information to the director. However, this potential detriment is assuaged by the requirement that, in order to be reported, perceptual content must be goal-related (e.g. of the same *type* as the blue box) (more on this later.)

3.3 Results Analysis

There are several advantages to employing the aspects of goal-oriented dynamic initiative. Goal-related percept reporting increases the amount of information available to the director at little cost (the time to process the utterance). It further grounds the decision-making process of the director in the world perceived by the searcher, increasing the body of shared knowledge. Goal-oriented

Table 2: In Dialog 1 the robot takes the initiative and reports entities in the environment that are appropriately similar to goal-related entities, demonstrating *goal-oriented percept reporting*

Dialog 1	Action / Perception
D: "Our task is to locate the blue box."	
R: "Acknowledged."	
D: "The room in front of you is called the breakroom."	
R: "Understood."	
D: "Is there a blue box in the breakroom?"	
R: "Hold on, let me check."	(drive, self, breakroom)
D: "OK."	
R: "No, but there is a yellow box in the breakroom."	percepts: yellow box
D: "OK, in that case go through the North door."	
R: "Roger."	

Dialog 2	Action / Perception
D: "Our task is to locate the blue box."	
R: "Acknowledged."	
D: "The room in front of you is called the breakroom."	
R: "Understood."	
D: "Is there a blue box in the breakroom?"	
R: "I do not know. I am not in the breakroom."	
D: "Go to the breakroom."	
R: "OK."	(drive, self, breakroom)
D: "Is there a blue box in the breakroom?"	
R: "No."	percepts: yellow box
D: "Is there a yellow box in the breakroom?"	
R: "Yes."	
D: "OK. Go through the North door."	
R: "Understood."	

action execution similarly decreases the number of exchanges in a dialog while goal completion reporting additionally streamlines dialog and, most importantly, provides objectively critical perceptual information to the director at the earliest opportunity. All three aspects build toward more natural HRI in that the robot presents itself as taking responsibility of its role in the task and having self-motivation for completing the task.

Potential disadvantages to this architecture include: (1) the supply of excess information to the interlocutor (as discussed in the previous section), (2) the possibility that the director may not expect or want the robot to act of its own initiative, and (3) the danger that the robot compromise the team's ability to complete the task by executing an action with negative consequences (despite mechanisms for lowering that probability).

Each of these potential disadvantages can be address with modifications/improvements to the architecture: (1) Consider again the example in Table 2. While it might be the case that there are

hundreds of boxes in the room and the robot incrementally overwhelms the interlocutor with what it deems to be goal-related information ("there is a yellow box, a green box, another yellow box, a red box ..."), it is straightforward to add mechanisms for reporting only the *most* goal-related percepts, or some other particular subset. Moreover a small adjustment to the natural language generation component would suffice in this case. Producing an utterance similar to "There are hundreds of boxes in the room, but not a blue one." would be much more appropriate; (2) Enabling the robot to adjust its degree of initiative-taking via a reinforcement learning component, or pre-configuring the architecture based on the director's preferred level of control; (3) As discussed previously, adding a variable to the *expected cost* calculation that accounts for any risk involved in executing an action. Again, a learning component would be useful in this case.

4 Conclusion

The DIDO architecture achieves more natural, effective HRI dialogues via each of its initiative-based features. It reduced the number of exchanges at several key points in a task-based dialog while also enabling the non-situated team member to act on percepts without making particular queries. These algorithms could be adjusted to work in similar task scenarios and there exists ample opportunity for improvement through future work. The main areas for improvements include policy acquisition (when to take initiative, which percepts to report, etc.) via learning and feedback mechanisms, methods for reducing the probability of taking high-risk actions, and deeper integration of leveraged situatedness for further reducing dialog complexity and tedium while increasing effectiveness and naturalness.

References

[1] Kathleen Eberhard, Hannele Nicholson, Sandra Kuebler, Susan Gundersen, and Matthias Scheutz. The indiana cooperative remote search task (crest) corpus. In *Proceedings of LREC 2010: Language Resources and Evaluation Conference*, Malta, May 2010.

[2] James Glass. Challenges for spoken dialogue systems. In *Proceedings of the 1999 IEEE ASRU Workshop*, 1999.

[3] Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, Donghyeon Lee, and Gary Geunbae Lee. Recent approaches to dialog management for spoken dialog systems. *Journal of Computing Science and Engineering*, 4(1):1–22, 2010.

[4] M. Scheutz. ADE - steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied Artificial Intelligence*, 20(4-5), 2006.