

PLANNING, OPTIMIZING, AND TAGGING:
THREE APPROACHES TO DIALOGUE MANAGEMENT

A PRELIMINARY EXAMINATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
OF UNIVERSITY OF COLORADO AT BOULDER
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Lee Becker
October 2009

© Copyright by Lee Becker 2010
All Rights Reserved

Contents

1	Introduction	1
1.1	Selected Literature	2
2	Dialogue as Planning	3
2.1	The BDI Architecture	3
2.2	Speech Acts and Illocutionary Force	3
2.3	TRAINS	4
2.3.1	Discourse Obligations	6
2.3.2	Mutual Belief and Grounding	7
2.4	Weighing in on the TRAINS approach	8
3	Dialogue as a Markov Decision Process	10
3.0.1	Reinforcement Learning for Dialogue Policy Design	11
3.0.2	Partially Observable Markov Decision Processes for Spoken Dialogue System	13
3.0.3	The challenge of scaling up	16
4	Learning Dialogue Structure	18
4.1	Dialogue Tree Structure	18
4.2	Learning Dialogue Structure	19
4.2.1	Utterance Analysis and Classification	19
4.2.2	Modeling Subtask Structure	20
4.3	Filling a void	21
5	Conclusions	22
A	Illocutionary acts used in TRAINS-93	24
	Bibliography	25

List of Tables

1.1	Sample dialogue	1
2.1	Examples of speech acts and their effects	4
2.2	Sample Obligation Rules [Traum, 1996]	6
2.3	Conversation Act Types [Traum, 1996]	7
3.1	Sample conversation with ASR misunderstanding from the CMU Communicator spoken dialog system	13

List of Figures

2.1	BDI Agent Architecture. [Traum, 1996]	4
2.2	A model of a simple deliberative agent. [Allen et al., 1995]	5
2.3	The TRAINS communicative agent model [Allen et al., 1995]	5
2.4	The TRAINS discourse actor algorithm [Traum, 1996]	8
3.1	NJFun state features and values [Singh et al., 2002]	12
3.2	POMDP Influence Diagram.	14
3.3	Spoken Dialogue Model [Williams and Young, 2007]	15
3.4	SDS-POMDP influence diagram[Williams and Young, 2007]	15
4.1	Dialogue Structure. [Bangalore et al., 2006]	19
4.2	Chunk Model Task Structure. [Bangalore et al., 2006]	20
4.3	Parse Model Task Structure. [Bangalore et al., 2006]	20

Chapter 1

Introduction

The real art of conversation is not only to say the right thing at the right place but to leave unsaid the wrong thing at the tempting moment. – Dorothy Neville

Over half a century has passed since Alan Turing posed a test of conversation as a measure of intelligence [Turing, 1950], however in that time not a single system has reliably passed this measure. While this can be seen as a failing of computer science and artificial intelligence, this probably better stands as a tribute to the complexity of human-to-human conversation.

- | | | |
|----|----------|---|
| 1 | Tutor: | Hi, how are you? |
| 2 | Student: | Good |
| 3 | Tutor: | Excellent |
| 4 | Tutor: | Let's talk a bit about your experiences with science.
Tell me about what you have been doing in science most recently. |
| 5 | Student: | We've been learning about circuits and how to work light bulbs |
| 6 | Tutor: | Circuits and working light bulbs, cool. Tell me more about circuits |
| 7 | Student: | Circuits are something that (uhm) it travels through wires which (uhm) |
| 8 | Tutor: | Very interesting. So, tell me more about what travels |
| 9 | Student: | Electricity |
| 10 | Tutor: | Very good. Lets look at this picture. What's going on here? |
| 11 | Student: | The battery is (uhm) making the electricity run through the- |
| 12 | Tutor: | Ok, so the battery makes the electricity, is that correct? |
| 13 | Student: | Yes |
| 14 | Tutor: | Good. Tell me more about how the electricity flows |

Table 1.1: Sample dialogue

As the Neville quote at the beginning of this chapter suggests, conversation is much more than saying the right thing. Consider the sample of tutorial dialogue shown in Table 1.1. The tutor must not only understand the student's utterances, but she must track and integrate them into an ever-changing context. Moreover she must not just understand an utterance in isolation, but also simultaneously make inferences based on parts from the entire conversation. Both parties must work to sustain the conversation, showing interest and providing feedback and acknowledgment when appropriate. On top of this already complex cognitive load, the speakers must work within the confines of social and cultural norms; for example, the student provides an answer when asked a

question. In this particular dialogue the challenge for the tutor is in deciding what is the appropriate response given everything a student has said. From a wider perspective, the tutor may be choosing her phrasing in hopes of carrying out a larger lesson plan and maximizing student learning. Every utterance in a conversation is a decision among several alternatives – rehash or advance the current subject matter, take or defer control of the conversation, clarify or wait for additional details, etc. . . .

In spoken dialogue systems (SDS) the decision making functionality described above is often encapsulated within a unit referred to as the *dialogue manager*. Typically, this module accepts a semantic representation of user utterances produced by a *natural language understanding* (NLU) unit, which is subsequently integrated and updated into a model of the discourse. Finally, at the end of this analysis, the dialogue manager will produce an appropriate response.

There are a wide number of techniques to perform dialogue management. Perhaps the first successful dialogue manager was ELIZA [Weizenbaum, 1966], which generated responses simply by transforming and regurgitating user utterances. Though ELIZA did not maintain any notion of dialogue context, its intentionally vague responses do a fairly convincing job of convincing users it really is a Rogerian therapist. Finite state automata (FSA) provide a simple means for incorporating context into dialogue management with each state corresponding to a prompt and each arc representing a possible user utterance. The major drawback to this approach is inability to stray from the set of predefined arcs. Frame-based approaches model context as a list of items to be prompted for, and can be more flexible than FSAs because filling an item does not require being in a specific dialogue state. While these approaches have been widely deployed in real-world systems, they require significant effort to build a robust, cogent dialogue application. The remainder of this paper will focus on approaches that attempt to minimize this effort through better modeling of dialogue state.

1.1 Selected Literature

The five papers selected in this survey represent three different approaches to dialogue management, each of which have made different breakthroughs towards the goal of attaining intelligent conversational agents. Two of the selections highlight how classical AI plan recognition and plan execution can be employed in a dialogue system ([Allen et al., 1995], [Traum, 1996]), while another two show how statistical machine learning methods and Markov Decision Processes provide a means for generating an optimal dialogue policy ([Singh et al., 2002], [Williams and Young, 2007]). The final selection illustrates how inferring dialogue structure from large corpora can bridge the gap between the two other approaches ([Bangalore et al., 2006]).

Chapter 2

Dialogue as Planning

Words are also actions, and actions are a kind of words. – Ralph Waldo Emerson

Though a dialogue manager can be viewed simply as a module that accepts language as input and generates language as output, it can be advantageous from a design standpoint to think of it less as a system and more as an agent – an entity with its own mental state and capacity for action and observation. Equipped with these states, actions and observations, an intelligent agent can reason about the world. By adding goal states, an agent can move from monitoring the world to becoming an active participant, developing and executing plans to achieve its goals. In a similar manner, a conversational agent equipped with mental states and goals, can take into account more complex dialogue context, and produce utterances that have relevance beyond the point at which they are generated.

2.1 The BDI Architecture

A common way to frame this agentic view of planning is within the BDI (belief, desire, and intention) architecture [Bratman et al., 1988]. In this architecture the agent maintains a representation of the state of the world (beliefs) and selects goals (intentions) to achieve based on a predetermined set of desires (a representation of how the agent would like the world to look). Once a goal is selected an agent can construct and subsequently execute the plan or sequence of actions used to achieve the desires. A plan carries with it expectations of how its actions will affect the world; performance and execution of plan-specified actions will create actual results. Taken together, expectations and results form the two ends of a process called *belief monitoring*. As actions are taken, the agent updates its beliefs based on perceptions that take into account both the plan expectations and resulting observations. A diagram of this process and the BDI architecture is shown in Figure 2.1.

2.2 Speech Acts and Illocutionary Force

For problems requiring action over physical domains mapping from the problem space to a BDI architecture is relatively straightforward. Conversely, the connection between BDI, planning, and dialogue systems is not immediately intuitive. Though we may plan our words and utterances carefully during the course of a conversation, their exact effect on the world is not as apparent. While an action like turning a key will directly result in opening a lock, the effect of a statement

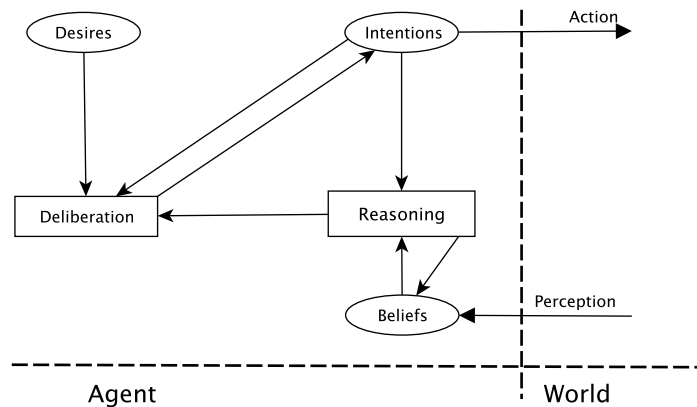


Figure 2.1: BDI Agent Architecture. [Traum, 1996]

like “*Did you take out the trash?*” seems indeterminate. Depending on the context and intent it could be perceived as a literal yes-no inquiry or it could be interpreted as a request.

Austin created the term *speech act* to describe how words and utterances can have an *illocutionary* force, i.e. the power to perform actions [Austin, 1962]. Searle subsequently noted how the propositional content of an utterance does not sufficiently describe its performative function [Searle, 1975]. Allen, Cohen and Perrault ([Cohen and Perrault, 1979], [Allen and Perrault, 1980]) provided the bridge between speech act theory, artificial intelligence, and planning. Their key finding was that speech acts can be expressed in terms of preconditions and effects relating to the mental state of the agents. Table 2.1 lists some examples of speech acts and their effects in propositional logic. Tying this back to the BDI agent architecture, a plan becomes a series of speech acts that are carried out by an agent, and perception relates closely to interpretation and understanding of spoken utterances.

REQUEST(speaker, hearer, act):
effect: speaker WANT hearer DO act
INFORM(speaker, hearer, proposition):
effect: KNOW(hearer, proposition)

Table 2.1: Examples of speech acts and their effects

2.3 TRAINS

The TRAINS System [Allen et al., 1995] is a conversational agent that follows in Allen, Cohen, and Perrault’s tradition of combining BDI-based planning with speech act theory. Although TRAINS is a logistics and planning application for coordinating transport and production of freight, it serves more as a vehicle to investigate issues of discourse processing and dialogue management. Rather than directly presenting knowledge of the world, TRAINS users interact with a problem-solving

assistant. Through conversation, the user works with the assistant to develop a mutually agreed upon plan of action to accomplish goals in the TRAINS domain. Implementation of TRAINS begins with a BDI-based, deliberative agent architecture (Figure 2.2). Replacing units from the deliberative architecture agent with ones more appropriate for speech and language processing yields the communicative agent architecture shown in Figure 2.3.

The TRAINS communicative agent model extends the notions of beliefs and desires to also encompass shared beliefs between conversational participants as well as obligations caused by the act of conversing. Because actions in the dialogue module are limited only to communicative acts, the TRAINS dialogue manager does not create a plan of action for its planning system, but rather creates speech acts whose natural language manifestation effectively carry out the agent’s plan. A separate BDI model captures the interactions between the system and agents who perform the actions in the simulated TRAINS world. Lastly, the process of interpreting observations finds a parallel in the tasks of natural language understanding and speech act recognition, where utterances from the manager (user) are translated into an appropriate belief-state representation. A full list of the TRAINS speech acts can be seen in Appendix A.

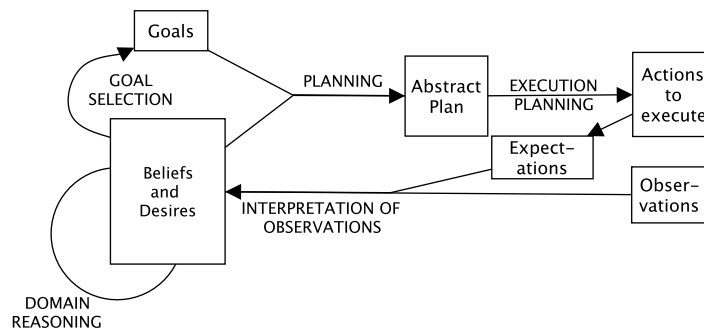


Figure 2.2: A model of a simple deliberative agent. [Allen et al., 1995]

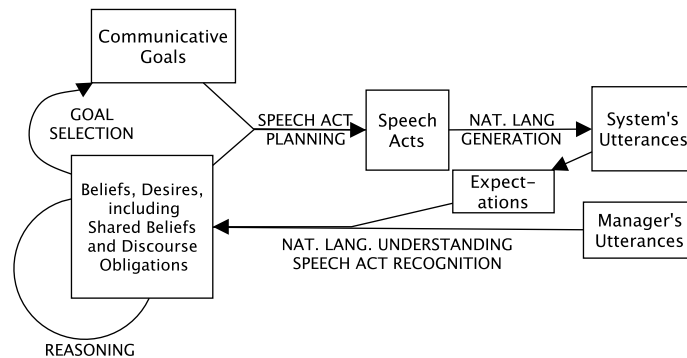


Figure 2.3: The TRAINS communicative agent model [Allen et al., 1995]

2.3.1 Discourse Obligations

Conversation does not revolve purely around forming a shared plan between dialogue partners or achieving some intention. It is expected in discourse for one person to answer another, even if the response does nothing to advance the goals for either party. Traum gives the example of a stranger asking “Do you know the time?” [Traum and Allen, 1994]. When dealing with strangers, there is no shared plan, and from a strictly strategic point of view, the responder has no stake in the stranger’s goals. In fact, answering will actually impede progress. But to not answer, would be considered unnatural, unconventional, and impolite.

In [Traum and Allen, 1994] and [Traum, 1996] Traum argues that belief, desires, and intentions do not fully explain what compels one speaker to answer prompts from another speaker, as they govern the possibility and utility of an action, but do not govern the permissibility of actions. Using a strictly BDI model, for the “Do you know the time?” scenario would require that the responder assume the goals of the stranger. To avoid this complexity in planning, Traum suggests an additional attitude known as *obligation* to account for the external pressures that arise from social interaction. Work in the TRAINS literature focuses specifically on *discourse obligations*, the set of conversational actions an agent *should do* according to social and conversational norms.

Like with speech acts theory, discourse obligations produce observable effects on the participants of the conversation. For example making a promise to do some task creates an obligation to get the task done. Even more subtly, a simple statement may require acknowledgment on behalf of the speaker. Table 2.2 highlights some of the discourse obligation rules used by TRAINS.

Source of obligation	Obligated action
S ₁ Accept or Promise A	S ₁ achieve A
S ₁ Request A	S ₂ address Request: accept or reject A
S ₁ Y/N-Question whether P	S ₂ Answer-if P
S ₁ Wh-Question	S ₂ Inform-ref x
utterance not understood or incorrect	repair utterance
S ₁ Initiate discourse unit	S ₂ acknowledge discourse unit
Request Repair of P	Repair P
Request Acknowledgement of P	acknowledge P

Table 2.2: Sample Obligation Rules [Traum, 1996]

Though discourse obligations simplify the planning process, adding them to the BDI paradigm introduces a tension in the decision making process. Even in a cooperative dialogue, there are cases where fulfilling an obligation may come into direct conflict with the dialogue agent’s immediate goals. There are a variety of strategies for prioritizing obligations relative to desires. The two most straightforward strategies include 1) performing all obligated actions and 2) performing only the actions that will result in a desired state. The TRAINS dialogue manager uses the former approach, dealing with obligations first, and in the absence of obligations, performing actions to satisfy its intentions. The pseudo-code for this behavior can be roughly characterized as:

```

loop
  if system has obligations then
    address obligations
  else if system has performable intentions then
    perform actions

```

```

else
  deliberate on goals
end if
end loop

```

This behavior also helps to create a mixed-initiative dialogue policy. While goal-driven actions usually indicate speaker-led initiative, obligation-driven actions reflect deferment of initiative to the opposite party. By accommodating discourse obligations, the TRAINS model provides a mechanism for enforcing system cooperation even when the system’s goals are in conflict with the user goals.

2.3.2 Mutual Belief and Grounding

Because a conversational agent does not act in isolation, it is important for its mental state representation to not only maintain a set of private beliefs about the world, but also a set of beliefs shared with other agents. Since dialogue in TRAINS is directly driven by the desire by both parties (system and user) to develop a mutually agreed upon plan, part of this mental representation is needed to model the domain plan-under-construction. The other part is needed to enable the process of building common understanding between the speakers to ensure that all parties can be confident that their contributions are factored into the body of the conversation. The process of maintaining a mutual understanding is commonly referred to as *grounding*, while the shared knowledge is called *mutual belief*. Traum considers the original speech act model to be insufficient for handling of mutual belief, as it considers mutual belief to be constructed directly from the statements made. They argue that language production is inherently noisy, and that mutual belief is constructed through:

“an elaborate process of feedback that can accompany initial utterances, and it is generally only after some sort of acknowledgment that the assumption of mutual belief is made.” [Traum, 1996]

Consequently, TRAINS extends the core speech acts to account for more complex behavior like grounding and turn-taking. Table 2.3 shows this extended set, which are referred to as conversation acts. These are used in conjunction with the discourse obligation rules in Table 2.2 to form the algorithm shown in Figure 2.4.

Act Type	Description	Sample Acts
Turn-taking	Maintain, release, or take the turn in the dialogue	take-turn, keep-turn
Grounding	Deal with establishing shared knowledge about the dialogue	repair, acknowledge
Core Speech Acts	The illocutionary acts	inform, yes-no-question, accept, request
Argumentation	Characterize the relationship between utterances	elaborate, Q&A

Table 2.3: Conversation Act Types [Traum, 1996]

```

1: while conversation is not finished do
2:   if system has obligations then
3:     address obligations
4:   else if system has turn then
5:     if system has intended conversation acts then
6:       call generator to produce NL utterances
7:     else if some material is ungrounded then
8:       address grounding situation
9:     else if some proposal is not accepted then
10:      consider proposals
11:    else if high-level goals are unsatisfied then
12:      address goals
13:    else
14:      release turn or attempt to end conversation
15:    end if
16:  else if no one has turn then
17:    take turn
18:  else if long pause then
19:    take turn
20:  end if
21: end while

```

Figure 2.4: The TRAINS discourse actor algorithm [Traum, 1996]

2.4 Weighing in on the TRAINS approach

Viewing the dialogue manager as an agent, enables the system to reason not only on the state of the world, but on the mental states of other agents, itself included. Furthermore, framing dialogue in this way allowed work on TRAINS to build from an already rich history in classical AI planning.

Perhaps the TRAINS project's lasting contribution, is their dissection of the dialogue management into several distinct, tenable problems including conversation act analysis, discourse obligation handling, and deliberation and planning. The approach taken by TRAINS at the time was considered quite ambitious, and it continues to be one of the only systems that attempts to fully capture the underlying linguistic phenomena driving the dynamics of dialogue. The use of speech act theory provides a concise description of conversational behavior, and the extending this theory to encompass discourse obligations and grounding acts, creates more dynamic and flexible conversations. Moreover, by adding different rules or effects, the TRAINS architecture could be used to compare different discourse theories against one another.

A major drawback to the planning approach to dialogue is its rigidity. Because plan-construction is a process of discovery, done by searching for the actions that will lead to the goal state, it may be difficult to alter the behavior of a dialogue agent. Though priorities and end-goals can be adjusted, there is little control over the intermediary actions leading to a goal state. Authoring the preconditions, causes, and effects of a rule may pose another challenge

. A rule-based approach to conversation act analysis may not sufficiently account for localized changes in context to infer the appropriate effect. In some cases, it may seem that selection of speech or conversation act is an over-commitment on behalf of the system.

Additionally the algorithms used for accounting for discourse obligations and mutual belief may be too-heavy handed. Though they do provide an important component to the overall flow of conversation, there is no way to use them judiciously. While most utterances do require some form acknowledgment, not all do, and to require this confirmation for every transaction can lead to user frustration. As will be shown in the following chapter, additional flexibility may be better achieved using a probabilistic approach.

Chapter 3

Dialogue as a Markov Decision Process

When one admits that nothing is certain, one must, I think, also admit that some things are much more nearly certain than others. – Bertrand Russell

Even when a dialogue system can fully understand a user’s goals and infer a plan of action, the best sequence for eliciting information may vary greatly from user to user. While some users may require a more explicit set of prompts, others may prefer to maintain more control of the dialogue flow. A distrusting or novice user may require constant verification of actions, while a more trusting user would need no confirmation. Building flexibility like this into a dialogue system can be a difficult, labor-intensive task. In practice, authoring a robust dialogue policy often requires an expert to specify appropriate actions for each state of the dialogue, consequently dialogue authorship becomes a constant trade-off between effort and robustness. Even when sufficient effort is taken to allow for flexibility, the conversations in a deployed system may not reflect those envisioned by the dialogue author, possibly necessitating an expensive cycle of refinement and testing. Handling noisy input from other components of the dialogue system like the NLU or ASR, can further complicate the process, as actions vary greatly if the system does not understand the user.

A common technique for dealing with uncertainty is to cast the problem in a probabilistic light. Markov Decision Processes (MDPs) provide a probabilistic framework for modeling planning and decision-making process over time. Central to the MDP framework is the Markov assumption, which says that future states depend only on the current state, and are independent of other past states. Thus, standard MDPs are an agent-based process defined by a 4-tuple consisting of:

1. S : A set of states describing the agent’s world
2. A : A set of actions the agent may take
3. T : A set of transition probabilities where $P_a(s, s') = P(s' | s, a) = P(s_{t+1} | s_t, a_t)$ is the probability that action a in state s at time t will transition to state s'
4. R : A set of rewards where $r(s, a)$ is the expected, immediate reward the agent receives for taking action a when in state s .

Assuming an infinite-horizon, i.e., there is no bound on when the reward can be obtained, the cumulative reward, Q is defined as:

$$Q = \sum_{t=0}^{\infty} \lambda^t r(s_t, a) \quad (3.1)$$

where λ is a discount factor $0 \leq \lambda \leq 1$. The Bellman equation capitalizes on the Markov assumption and defines the cumulative reward recursively as:

$$Q = r(s, a) + \lambda \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a') \quad (3.2)$$

Overall, the motivation for using the MDP architecture is to find the policy (mapping of states to actions) that will yield the highest possible cumulative reward. This policy is defined as the function $\pi(s)$. When applied to the dialogue management problem, the policy should yield a set of dialogue actions (utterances, statements, etc...) that will best match to the user and conversation state. There are a variety of techniques for calculating the optimal policy including value iteration, policy iteration, and Q-learning depending on the exact parameters of the MDP. A full explanation of these approaches is beyond the scope of this review, and the reader should refer to [Russell and Norvig, 2003] for additional information.

The primary challenge in using MDP mechanisms lies in the selection of the relevant features to encode the state space and choosing appropriate rewards. State transition and reward probabilities are typically computed from a corpus of dialogues annotated with states and actions. In practice, however, obtaining enough data to sufficiently cover all possible state spaces is expensive, noisy, or impossible. There are two main approaches for dealing with this limitation. One is to reduce the state-space, so that random exploration can provide enough coverage to obtain transition and reward probabilities. The other approach is to automatically produce a corpus from millions of interactions conducted with a simulated user.

3.0.1 Reinforcement Learning for Dialogue Policy Design

[Singh et al., 2002] utilized reinforcement learning, an extension of the MDP, to build an optimal dialogue policy for the NJFun System, a real-time spoken dialogue system that provides users with information about things to do in New Jersey. This was among the first works to build a dialogue policy using data from real users.

Instead of using reinforcement learning to optimize over all decisions, [Singh et al., 2002] focused on optimizing only on two types of decisions: 1) initiative and 2) confirmation. The initiative problem refers to the task of choosing to what degree the user controls the conversation. In the NJFun system this comes down to how open or direct to state information seeking prompts. An imbalance in selecting initiative can lead to dialogues that are either too restrictive or too unstructured. Confirmation refers to whether or not to verify a requested attributes value. Overuse of confirmation can lead to user frustration, while lack of confirmation may lead the dialogue down a wrong path.

Though it is possible to encode nearly any feature into the state-space used by an MDP, [Singh et al., 2002] opted to use what they refer to as a state-space estimator to compress the state, using only a small subset of all potential features (Figure 3.1). The authors of NJFun felt that this state-space is small enough to “support the construction of an MDP model that is not sparse with respect to S , even using limited training data”. The state-space estimator feature set yields 62

possible states. Of these, 42 are choice-states with 2 actions per state (i.e. confirm / not confirm, system-initiative / user-initiative), which in turn can produce 2^{42} unique dialogue trajectories.

Feature	Values	Explanation
Greet(G)	0,1	Whether the system has greeted the user
Attribute(A)	1,2,3,4	Which attribute is being worked on
Confidence / Confirmed (C)	0,1,2,3,4	0,1,2 for low, medium and high ASR confidence. 3,4 for explicitly confirmed and disconfirmed
Value(V)	0,1	Whether the value has been obtained for current attribute
Tries(T)	0,1,2	How many times current attribute has been asked
Grammar(M)	0,1	Whether non-restrictive or restrictive grammar was used
History(H)	0,1	Whether there was trouble on any previous attribute

Figure 3.1: NJFun state features and values [Singh et al., 2002]

Finding an Optimal Policy

Because the tasks and interactions in the NJFun system were well understood, and because there were only 62 possible dialogue states, [Singh et al., 2002] were able to employ the technique of random exploration to collect the training data necessary for a reinforcement learner to compute the reward and transition values to produce a (near) optimal dialogue policy.

For each state requiring a choice of initiative or confirmation, the action taken was randomized. This random-policy equipped system was then deployed to 54 users. Each user was given six different application-tasks, resulting in 311 free-form dialogues to be used for training. The reward measure used was binary task completion. Scores of +1 were given to dialogues that queried the database for the exact set of attributes (activity type, location, time of day) associated with a task, otherwise dialogues were given a score of -1. Finally with the set of states, actions, and rewards in place, an optimal dialogue policy was computed using reinforcement learning. The learned policy found “the optimal use of initiative is to begin with user initiative, then back off to either mixed or system initiative”. For confirmation “the optimal policy is to mainly confirm at lower confidence (ASR) values”, however other features play a role making for a more complex handling of confirmation.

Next the NJFun system was reimplemented using the decisions prescribed by the optimal policy. To evaluate the effectiveness of this policy, an additional 21 users interacted with the system yielding 124 testing dialogues. When comparing the learned policy versus the trained policy, the authors of NJFun found the learned policy did not significantly outperform ($p = 0.059$) the baseline when evaluated on a strict binary completion measure. However, they did find statistical improvement in the weak completion ¹ and ASR ² measures. Unlike the binary completion measure, the reinforcement learning did not optimize for either of these measures.

¹Weak completion gives partial credit for terms queried from the binary completion measure

²ASR is another variation on the binary completion measure giving a score of 1 for each correct attribute value and 0.5 for every wildcard

3.0.2 Partially Observable Markov Decision Processes for Spoken Dialogue System

A major limitation of MDPs is their assumption that the current state of the dialogue reflects the exact state of the world (dialogue). Even though they do provide a mechanism for optimizing over uncertainties like speech recognition errors, they can really only deal with this uncertainty at the time of the decision. If an error goes undetected, the dialogue manager will continue along the same path and drift further away from the user’s original intent. A conversation illustrating such behavior is shown in Table 3.1. In the first turn the system asks for the user’s name, but bad ASR output produces the phrase “Athens in Akron”. Because two of the words in the utterance are place names, the system then assumes that the user is taking the dialogue initiative, and is specifying venues instead of trying to confirm the name. While confirmation of the name would have been the best action at turn 5, the features like ASR confidence values and dialogue history may have indicated otherwise. A potentially more robust mechanism would maintain a parallel hypotheses, and backpedal or switch strategies when a hypothesis becomes sufficiently false. This parallel hypothesis is the underlying intuition when modeling dialogue with a Partially Observable Markov Decision Process (POMDP).

Turn	Turn-taker	Utterance [ASR Output]
1	S	Are you a registered user?
2	U	No, I’m not [NO I’M NOT]
3	S	What is your full name
4	U	⟨Username⟩ [ATHENS IN AKRON]
5	S	There is an Athens in Georgia and in Greece. What destination did you want?
6	U	Neither [NEITHER]
7	S	I’m sorry, I did not catch that. There is an Athens in Georgia and in Greece. Which destination do you want?
8	U	Georgia [GEORGIA]
9	S	A flight from Athens... Where do you want to go?
10	U	Start over [START OVER]

Table 3.1: Sample conversation with ASR misunderstanding from the CMU Communicator spoken dialog system

POMDPs maintains the transition model and reward from functions from MDPs, but it also extends the MDP architecture by adding a set of observations, O the agent can receive about the world; observation probabilities Z , and a belief state $b(s)$. Because POMDPs strive to account for multiple possible interpretations of the current state, an exact value of state does not exist in the model. Instead, at a given time-step, the POMDP will maintain a probability distribution over all possible states known as the “belief state”, b . Thusly, the expression $b(s)$ refers to the probability of being in state s . Whereas the MDP could use the state to accurately reflect what the agent perceived, the probabilistic account of state in the POMDP architecture dictates the need for O , a set of observations the agent can receive about the world as well as an observation function Z to define probabilities of observation $P(o'|s', a)$. At each time-step the POMDP agent receives observation $o' \in O$ and performs “belief monitoring” using the following:

$$b'(s') = \frac{Z \cdot \sum_{s \in S} T \cdot b(s)}{p(o'|a, b)} = \frac{p(o'|s', a) \cdot \sum_{s \in S} p(s'|a, s)b(s)}{p(o'|a, b)} \tag{3.3}$$

$$= k \cdot p(o'|s', a) \sum_{s \in S} p(s'|a, s)b(s) \tag{3.4}$$

The relations between states, observations, actions, rewards, and time is shown graphically in Figure 3.2.

Like with the MDP, the goal is to find a policy that maximizes the expected return on the cumulative reward function 3.1. While the MDP policy $\pi(s)$ was a direct mapping of states to actions, the POMDP policy $\pi(b)$ maps from belief state to actions.

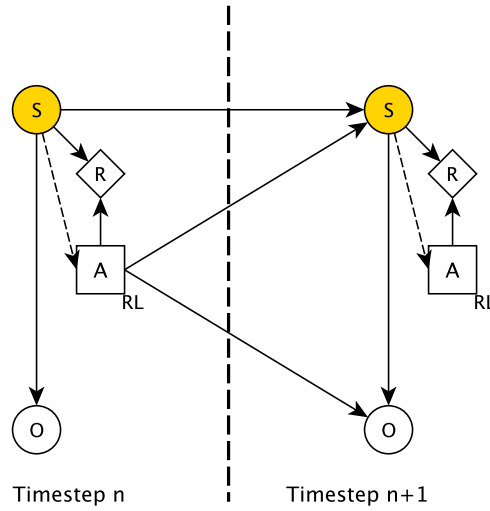


Figure 3.2: POMDP Influence Diagram.

Circles represent random variables. Squares represent decision nodes. Diamonds represent utility nodes. Shading indicates unobservable variables. Solid arcs indicate causal effect, while dashed arcs indicate a distribution or estimate is used (as opposed to an unobserved value). RL indicates that reinforcement learning is used to choose actions.[Williams and Young, 2007]

When inspecting a model of a spoken dialogue system like the one shown in Figure 3.3, it is important to note that the agent’s dialogue model never observes the actual intentions and actions produced by the user. Instead the dialogue manager receives a proxy for these by way of the noisier speech recognition and language understanding modules. Williams and Young recognized this inherent uncertainty and adapted the POMDP architecture to spoken dialogue systems by modeling the actual (but partially unobservable) state, S , as a tuple of the user’s goal, s_u , the corresponding action (a.k.a. intention) taken to achieve the goal, a_u , and the history of utterance, s_d , hence:

$$s = (s_u, a_u, s_d) \tag{3.5}$$

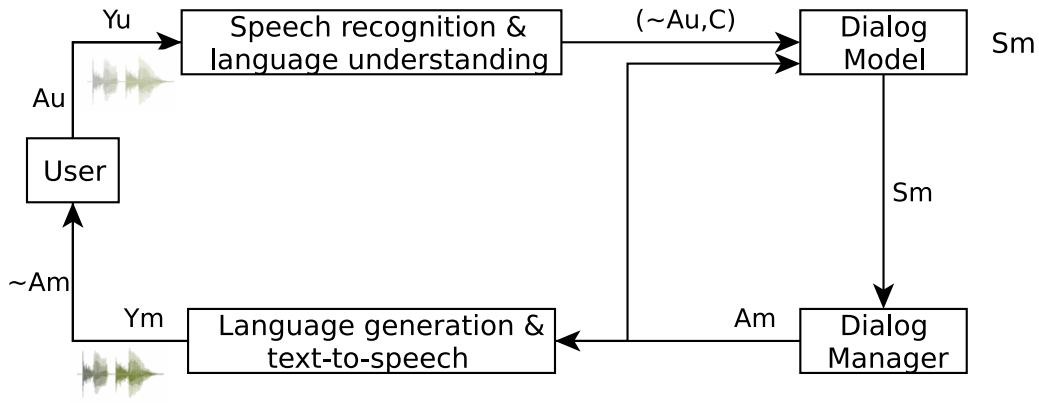


Figure 3.3: Spoken Dialogue Model [Williams and Young, 2007]

and the state captured within the dialogue model s_m represents our belief over the state, i.e,

$$s_m = b(s) = b(s_u, a_u, s_d) \tag{3.6}$$

Lastly, the observable quantities are represented by the output from the ASR, \tilde{a}_u , and its corresponding confidence score c . A graphical representation of the [Williams and Young, 2007]’s spoken dialogue system POMDP (SDS-POMDP) is shown in Figure 3.4.

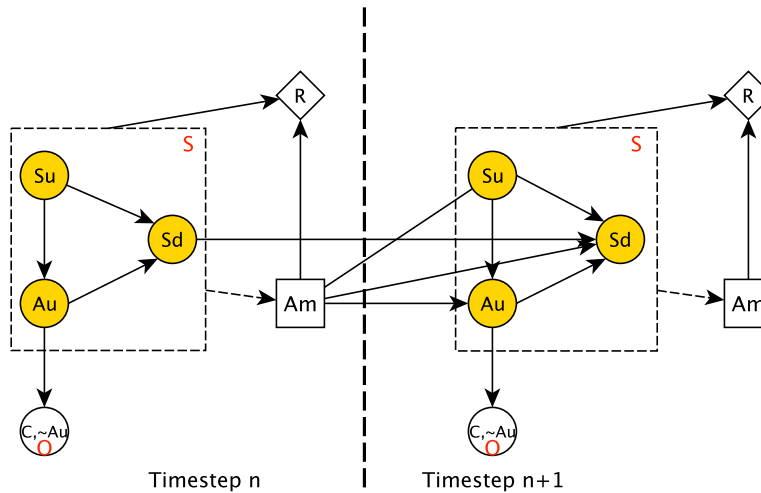


Figure 3.4: SDS-POMDP influence diagram [Williams and Young, 2007]

The two central claims made by [Williams and Young, 2007] are that POMDPs perform better because they 1) maintain parallel dialogue state hypotheses and 2) incorporate confidence scores directly into the belief state update. To test this hypothesis, [Williams and Young, 2007] created a test-bed simulation in the travel domain. Users are given the goal of buying a ticket from one city to

another. The system asks users a series of questions and then finalizes the ticket purchase. In total the system has 16 available actions including *greet*, *ask-from/ask-to*, *confirm-to-x/confirm-from-x*, *submit-x-y*. Additionally, the system has the option of abandoning the dialogue via the *fail* action. Combining these actions with user actions, yielded a total of 1945 dialogue states.

Because the observation function $p(\tilde{a}'_u, c'|a'_u)$ is impossible to estimate directly from the data, the approximation used in the travel domain simulation is:

$$p(\tilde{a}'_u|a'_u) = \begin{cases} p_h(c') \cdot (1 - p_{err}) & \text{if } \tilde{a}'_u = a'_u \\ p_h(1 - c') \cdot \frac{p_{err}}{|A_u|} & \text{if } \tilde{a}'_u \neq a'_u \end{cases} \quad (3.7)$$

where c' is the speech recognition confidence over the interval $[0,1]$, and $p_h(c')$ is an exponential probability density function with slope determined by parameter h , while p_{err} reflects the assumption that all confusions are equally likely.

$$p(\tilde{a}'_u|a'_u) = \begin{cases} p_h(c') \cdot (1 - p_{err}) & \text{if } \tilde{a}'_u = a'_u \\ p_h(1 - c') \cdot \frac{p_{err}}{|A_u|} & \text{if } \tilde{a}'_u \neq a'_u \end{cases} \quad (3.8)$$

The reward function used in the simulation attempts to reward for task completion and dialog “appropriateness”. An action receives a value of -3 for confirming a field before the user has referenced it; taking the *fail* action results in a -5 reward; a score of +10 is given for issuing the correct *submit-x-y* purchase query, and a score of -10 is given for issuing it incorrectly; all other actions are given values of -1. This reward schema reflects the notion that it is better to give up on a conversation early than to waste the user’s time on a fruitless series of interactions. With all the POMDP parameters established, optimization of was done using a variant of point-based value iteration. For comparison, a 51-state MDP, which used confidence value thresholds, was created for the same domain, and an optimal MDP policy was computed via Q-Learning. The experiment found that the POMDP obtained significantly higher cumulative returns across a variety of values for p_{err} and c . [Williams and Young, 2007] also found the POMDP policy yielded higher returns against a variety of hand-crafted policies. In another experiment, a POMDP model was trained and optimized using real dialogue data instead of a hand-crafted user simulation. Testing this model on a held-out set of dialogue 10,000 dialogue turns, found performance on the model optimized from simulated data closely tracked the model optimized from real data, indicating that “errors in the estimation of the user model can be tolerated”.

3.0.3 The challenge of scaling up

Markov Decision Processes and their derivatives like Reinforcement Learning and Partially Observable Markov Decision Processes provide a well-founded framework for modeling the agentive approach to dialogue management. Unlike the classical AI planning approach to dialogue management, the probabilistic underpinnings provide an elegant means for working with common sources of dialogue uncertainty like speech recognition or natural language understanding errors. The systems developed in [Singh et al., 2002] and [Williams and Young, 2007] provide examples of how optimizing an MDP can produce a dialogue policies that are more robust than those carefully created by domain experts.

Among the most attractive aspects of the MDP architecture is the flexibility to model or incorporate knowledge from any number of diverse sources. In a complex domain like tutoring systems,

one could imagine a very rich state representation that includes data on number of questions correctly answered and past test scores in addition to the more common dialogue state features already in common use. The POMDP architecture and its incorporation of hidden states provides a statistical machine learning based corollary for the BDI states so prevalent in the planning literature. In an ideal case, a dialogue manager would be able to include several unobservable features like emotional affect, user comprehension levels, and other intangibles into its state representation.

On the other hand, MDPs still have several shortcomings that prevent them from being deployed in more complex systems. Much of the success [Singh et al., 2002] and [Williams and Young, 2007] had with their systems could not be had with larger, non-trivial applications. In general MDPs do not scale well to a large number of states. Even worse, POMDPs suffer additional issues of intractability because of the need to model belief states for all possible values. In a system like the travel domain where the number of choices for destination cities could number in the thousands or more, this becomes too expensive to be practical.

Computational complexity aside, there remains the major issue of obtaining sufficient training data. While, [Singh et al., 2002] consider the approximately 300 dialogues used to compute an optimal policy to be a relatively small number, they benefited from having an application that required no more than 15 dialogue turns. Moreover, the nature of their utterances were well understood and easy to ascertain. Again, this approach would not scale to a more difficult task. [Williams and Young, 2007] circumvented this problem using user simulations, however this approach seems counter-intuitive. One could argue that the knowledge and effort required to create a realistic user simulation could be encoded directly into the dialogue policies. Learning theory suggests that at best an MDP would learn the rules defining the simulation behavior, and at worst learning something completely different.

Cumulative reward provides a consistent, quantitative method for evaluating policies versus one another, but choosing a reward function may not always be straightforward. For most task-oriented dialogue the reward function is calculated based on time to task completion. While this may be a sufficient proxy, it may not reflect intangibles like user satisfaction or quality of information presented. Selecting a reward for an application like intelligent tutoring systems is even less obvious. Users of these systems may not have a specific agenda, and the system designers hope to improve a difficult to measure value like learning gains.

Although, there is a strong desire for optimal policies, perhaps the key to making MDPs more generally useful for spoken dialogue systems would be to develop techniques that find sub-optimal, but still robust policies. As few conversations in reality are optimal, but most are resilient to errors and misunderstandings.

Chapter 4

Learning Dialogue Structure

Every discourse, even a poetic or oracular sentence, carries with it a system of rules for producing analogous things and thus an outline of methodology. – Jacques Derrida

Much of the work in dialogue management has focused expressly on choosing an appropriate response in the context of a dialogue. Both the AI plan recognition and Markov Decision Process techniques center on finding an optimal policy. Though these approaches have found ways of automating the dialogue planning process, they do little to address the effort required to create the states and actions on which a dialogue system is built. Moreover, these optimized policies are very much tailored to a specific task, and do not address the global similarities from dialogue to dialogue or from task to task. Conversely, [Bangalore et al., 2006] approach the problem less from an optimization perspective and more from a characterization viewpoint as part of a greater effort to build components for spoken dialogue systems using data-driven techniques.

4.1 Dialogue Tree Structure

[Bangalore et al., 2006] endeavored to learn models for predicting dialogue acts and subtask structures of dialogues from several corpora of task-driven human-human dialogues. Their approach was largely guided by the view that dialogue is a process of incrementally building a shared plan, which can be represented as a tree with hierarchical relationships between tasks, dialogue acts, utterances, and clauses (Figure 4.1). In the course of a dialogue, assigning a new utterance to the appropriate level of hierarchy becomes a problem of incremental parsing. Like how an incremental syntactic parser must decide if a new word starts a new phrase or extends an existing one, a dialogue parser must decide where to place a new utterance in the hierarchy. An utterance could signal the start of a new topic, or it may continue an existing one. Knowing how an utterance fits into the overall dialogue structure can then in turn be used to inform the dialogue manager of the current dialogue state and potential next actions. Another advantage in using incremental techniques is its efficacy in handling the real-time demands of spoken dialogue.

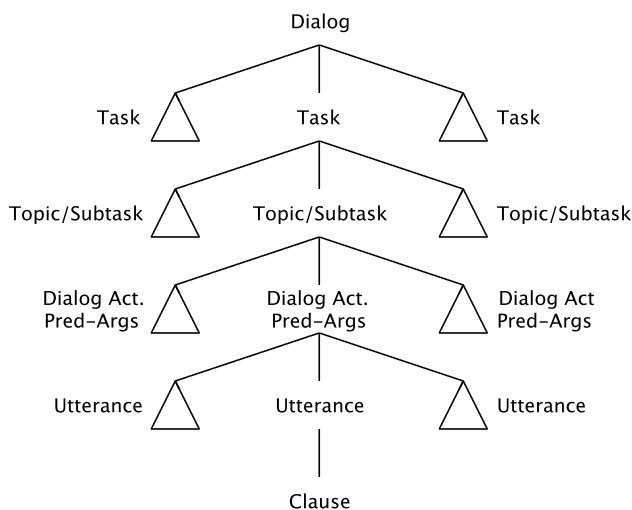


Figure 4.1: Dialogue Structure. [Bangalore et al., 2006]

4.2 Learning Dialogue Structure

4.2.1 Utterance Analysis and Classification

Building a model of dialogue structure required functional units to address each level of the hierarchy in Figure 4.1. At the lowest level an utterance segmenter was built to split spoken utterances into individual clauses and to clean up artifacts of speech like repair and disfluencies. This enables linguistic analysis of the utterances further up the hierarchy. Syntactic annotation was done first using a supertagger [Bangalore and Joshi, 1999] to capture local lexical dependencies within a clause. Supertags were then assembled into a full parse using substitution and adjunction operations from Tree-Adjoining Grammars (TAG) [Joshi, 1987], which in turn provide a means for deriving a dependency analysis of an utterance as well as its predicate-argument structure.

Understanding the interactions at the discourse level dictates the need for a dialogue act tagger. [Bangalore et al., 2006] trained their dialogue act classifier using several corpora annotated with dialogue acts. Annotation schemes like DAMSL [Core, 1998] were avoided because they are too general to provide the discriminatory power necessary for reliable learning and labeling. Instead, [Bangalore et al., 2006] used dialogue acts specific to their data and domain, namely customer-service dialogues related to the task of ordering products from a catalog. Example types of dialogue acts include *ask*, *explain*, *conversational*, and, *request* while example sub-types include *info*, *order_info*, *product_info*, *hello*, *thanks*, *repeat*, *order_status*, and more. Features extracted to train the dialogue act classifier were speaker information (e.g. instruct-giver versus instruct-follower), word trigrams from current and previous utterances, and the supertagged utterance. Error rates on the dialogue classifier outperformed systems referenced in previous literature.

4.2.2 Modeling Subtask Structure

While dialogue acts and utterances hint at the current conversational context, knowing an utterances relationships to the rest of the dialogue provides a critical piece of information for selecting among potential next actions. In particular a dialogue manager would be interested in knowing if the current utterance starts a new sub-task or continues one already in progress. In language applications sequence/sub-sequence determination can often be encapsulated in a chunk-model whereas a stricter structure determination implies some kind of parse-model. Figure 4.2 shows how a chunk model captures dialog act precedence relationships for an order placement task in the catalog domain, while Figure 4.3 shows the parse model’s richer structure that also includes the dominance relationships between dialog acts.

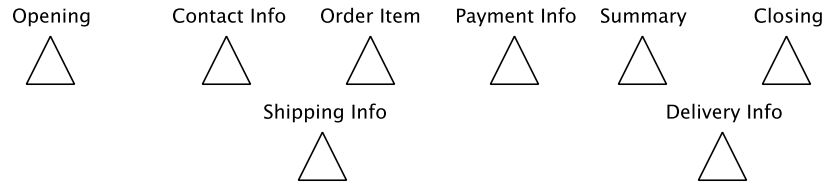


Figure 4.2: Chunk Model Task Structure. [Bangalore et al., 2006]

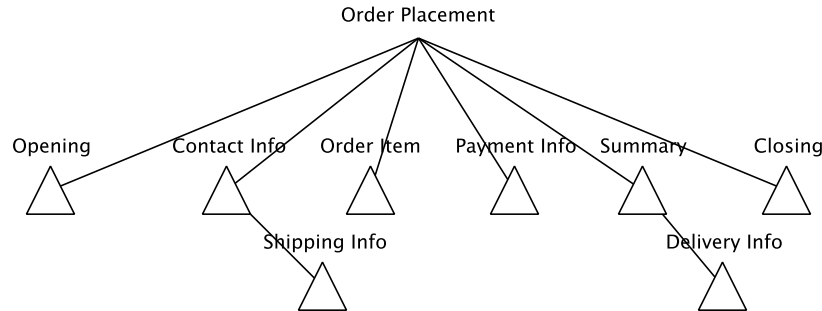


Figure 4.3: Parse Model Task Structure. [Bangalore et al., 2006]

The chunk-model used in [Bangalore et al., 2006] was essentially a BIO (beginning, inside, outside) sequence classifier adapted to dialogue subtask identification. Given a sequence of utterances $U = u_1, u_2, \dots, u_n$, the goal is to find the best subtask label $ST = \{st_B, st_I, st_O\}$ such that:

$$ST^* = \operatorname{argmax}_{ST} P(ST|U) \quad (4.1)$$

The parse-model attempts to incorporate additional structure by finding the most likely plan

tree, PT , given the sequence of utterances:

$$PT^* = \underset{PT}{\operatorname{argmax}} P(PT|U) \quad (4.2)$$

[Bangalore et al., 2006] utilized a top-down incremental parser that incorporates bottom-up information from the one-best chunk output. Evaluation of these models found performance on subtask sequence prediction to be comparable between the two models. Considered in the context of dialogue management where real-time latencies are a critical factor, the benefits of the chunk-model’s efficiency and capability for incremental segmentation outweigh any potential gain in accuracy obtained from the parse-model. Moreover, the additional structure from the parse-model provides little extra information over the purely sequential information found in the chunk-model.

4.3 Filling a void

Unlike the decision oriented techniques, the work in [Bangalore et al., 2006] does not suffer from the issues of scalability or tractability as there is no need to optimize over a large state space. This kind of approach seems well suited for rapid prototype of future dialogue systems. However the expense of collecting and annotating large corpora of dialogues may quickly impede progress. Additionally, it still remains to be seen if the models derived for one set of corpora will apply to other task-driven applications, especially considering the need for specialized, domain specific dialogue act tags. That said, this work’s goals of moving towards construction of a spoken dialogue system via data-driven techniques fills a gap, that has been sorely neglected by the dialogue community. While the work with MDPs are also data-driven, they do so more with an emphasis on optimizing individual decisions. In theory, learning a global structure for the dialogue may help in building an initial dialogue system that can be optimized later.

Chapter 5

Conclusions

We demand rigidly defined areas of doubt and uncertainty. – Douglas Adams

This paper has reviewed a wide range of techniques that address all or part of the challenge of building a dialogue manager. Though these techniques appear to be fairly disparate and unrelated, they are all attempting to exploit the structure and constraints of conversation to task-oriented dialogue.

An ideal dialogue manager would be able to integrate components from each of these approaches. The extended BDI-architecture described in [Allen et al., 1995] could provide a basic scaffold of the requisite functional units for building a dialogue manager. Though slightly inflexible, the rule-based approaches to discourse obligation handling and grounding operations coupled with the planning behavior of their conversation act theory could provide a suitable, baseline set of behaviors in the absence of additional context or data.

With infinite computing power and infinite training data, the full mental-space representation could be encoded into a POMDP, yielding an optimal policy that performs all of the duties related to the conversational task while still incorporating other meta-linguistic functionality that maintains a natural pace and flow of dialogue. For an application like tutorial dialogue, POMDPs could provide a powerful tool for modeling the student’s current level of understanding, especially with respect to other variables of their utterances. Unfortunately, the state space explosion makes this intractable. Realistically, MDPs and POMDPs may be best reserved for optimization of one-off variables like confirmation, acknowledgment, or repair-suggestion. From an evaluation perspective, MDP-based techniques may be the suited for comparing performance between systems and/or algorithms because of empirical quality of the reward function, however this does not eliminate the difficulty in finding a reward function that actually correlates to more natural dialogue.

Data-driven approaches like those used by [Bangalore et al., 2006] can potentially form a compromise between the two approaches mentioned above. Learning dialogue structure from corpora may lead to more flexible interactions than classical AI planning mechanisms alone, as it could be used to highlight the degrees of freedom in the dialogue flow. Data-derived task and sub-task structure could also minimize the upfront time spent authoring and designing a dialogue. Even without considering dialogue structure at-large, the same data could also be used to create a statistical conversation act classifier, which could in turn supplement any of the systems above. Most importantly the data-driven approach provides a way to incorporate syntactic and semantic information content into the dialogue decision making process. Because speech act theory does not address the propositional content of the utterance, dialogue act annotation schemes like DAMSL

[Core, 1998], provide very little information about what is actually occurring in the dialogue. By merging conversational acts with semantic content, there may be an opportunity to understand how entities and relationships affect the structure of the conversation.

Appendix A

Illocutionary acts used in TRAINS-93

Speech Act	Description
T-INFORM	The speaker aims to establish a shared belief in the proposition asserted
T-YNQ	The speaker asks a yes-no question, creating an obligation for the hearer to respond
T-CHECK	The speaker is verifying that a certain proposition is true.
T-SUGGEST	The speaker proposes a new item (action, proposition) as part of the plan
T-REQUEST	The speaker aims to get the hearer to perform some action. In the TRAINS domain, this is treated like a suggest, with the addition of an obligation on the hearer to respond.
T-ACCEPT	The speaker agrees to a prior proposal by the hearer.
T-REJECT	The speaker rejects a prior proposal by the hearer.
T-SUPP-INF	The speaker provides additional information that augments or helps the hearer interpret some other accompanying speech act.

Bibliography

- [Allen and Perrault, 1980] Allen, J. F. and Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178.
- [Allen et al., 1995] Allen, J. F., Schubert, L. K., Ferguson, G., Heeman, P., Hwang, C. H., Kato, T., Light, M., Martin, N. G., Miller, B. W., Poesio, M., and Traum, D. R. (1995). The trains project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical AI*, 7.
- [Austin, 1962] Austin, J. (1962). *How to Do Things with Words*. Harvard University Press.
- [Bangalore et al., 2006] Bangalore, S., Di Fabbrizio, G., and Stent, A. (2006). Learning the structure of task-driven human-human dialogs. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 201–208, Morristown, NJ, USA. Association for Computational Linguistics.
- [Bangalore and Joshi, 1999] Bangalore, S. and Joshi, A. K. (1999). Supertagging: an approach to almost parsing. *Comput. Linguist.*, 25(2):237–265.
- [Bratman et al., 1988] Bratman, M. E., Israel, D., and Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.
- [Cohen and Perrault, 1979] Cohen, P. R. and Perrault, C. R. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212.
- [Core, 1998] Core, M. G. (1998). Analyzing and predicting patterns of damsl utterance tags. In *In Working Notes AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 18–24.
- [Joshi, 1987] Joshi, A. K. (1987). Introduction to tree adjoining grammars. In Manaster-Ramer, A., editor, *Mathematics of Language*. John Benjamins, Amsterdam.
- [Russell and Norvig, 2003] Russell, S. and Norvig, P. (2003). *Artificial Intelligence A Modern Approach*. Pearson Education, Inc.
- [Searle, 1975] Searle, J. R. (1975). *A Taxonomy of Illocutionary Acts*.
- [Singh et al., 2002] Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

- [Traum, 1996] Traum, D. R. (1996). Conversational agency: The trains-93 dialogue manager. In *In Susann LuperFoy, Anton Nijholt, and Gert Veldhuijzen van Zanten, editors, Proceedings of Twente Workshop on Language Technology, TWLT-II*, pages 1–11.
- [Traum and Allen, 1994] Traum, D. R. and Allen, J. F. (1994). Discourse obligations in dialogue processing. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 1–8.
- [Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX(236):433–460.
- [Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.
- [Williams and Young, 2007] Williams, J. D. and Young, S. (2007). Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, 21(2):393–422.