

## Chapter 2

# Dialogue System Theory

A spoken dialogue system has a large number of complex problems to overcome. To simplify matters, two key assumptions are almost always taken. First, only dialogues with exactly two participants are considered and second, all interactions between the system and the user are in the form of turns. A *turn* in a dialogue is a period in which one of the participants has a chance to say something to the other participant.

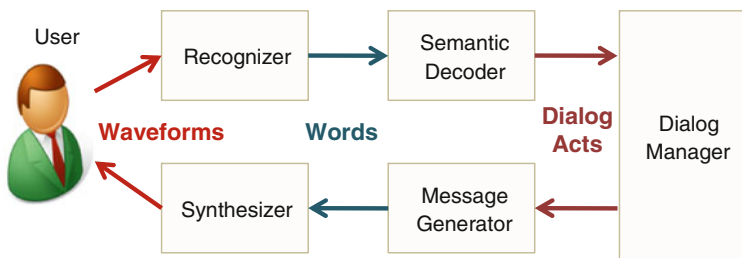
Under these assumptions, the dialogue will follow a cycle, known as the *dialogue cycle*. One of the participants says something, this is interpreted by the listener, who makes a decision about how to respond and the response is conveyed back to the original participant. This same process of listening, understanding, deciding and responding then occurs on the other side and the cycle repeats. Hence any dialogue system requires a number of components: one that can understand the user's speech, one that makes decisions and one that produces the system's speech.

Figure 2.1 shows a graphical representation of the components which achieve these goals and how they relate to one another. Details of the various methods for developing these components will be discussed in Sect. 2.1. Although these components are all that is required for building a spoken dialogue system, it is useful to also have a simulation environment for testing and training purposes. This is provided by a user simulator discussed in Sect. 2.2. This thesis is concerned mainly with the decision making component of a spoken dialogue system, known as the *dialogue manager*. Section 2.3 will discuss various paradigms for building the dialogue manager and provide a literature review of dialogue management theory.

## 2.1 Components of a Spoken Dialogue System

### 2.1.1 Speech Recognition

The first task of a spoken dialogue system is to understand the user's speech. This initial stage begins by using a *speech recognition engine* to convert the audio input



**Fig. 2.1** A graphical representation of the dialogue cycle

into text. Once the text form is obtained it is analysed to identify the underlying meaning.

State-of-the-art speech-recognition methods use a statistical model called the Hidden Markov Model (HMM) to estimate the most probable sequence of words for a given audio stream. This component of a dialogue system may be built using various publicly available toolkits. Examples include the ATK/HTK (Young et al. 2000) and SPHINX packages (Walker et al. 2004). Commercial systems are also available and may be obtained from companies such as Nuance, Loquendo and IBM.

The performance of the speech recognition engine will depend greatly on the difficulty of the task and on the amount of in-domain training data. In the case of limited-domain dialogue systems where the user is allowed to speak freely (i.e. they are not constrained by being asked to select from a list), high error rates are common. Raux et al. (2006), describes the Let's Go! bus information system, which has a sentence average word error rate of 64 %. The word error rate of the ITSPPOKE tutorial system is 34.3 % (Litman et al. 2006).

Most current spoken dialogue systems use only the most likely hypothesis of the user's speech. State-of-the-art recognisers can, however, output a list of hypotheses along with associated confidence scores. This list is called an *N-best list*, where *N* denotes the number of hypotheses. The confidence scores give an indication of the likelihood that the recogniser attaches to each word sequence. Ideally these confidence scores will give the posterior probability of the word sequence given the audio input (Jiang 2005). In some cases the recogniser may also return a *word-lattice* to represent the set of possible hypotheses. Such word-lattices may be converted into N-best lists by first converting the lattice to a confusion network and then using dynamic programming to find the minimum sum of word-level posteriors (Evermann and Woodland 2000). Further details on confidence scoring issues are provided in Appendix F.

### 2.1.2 Spoken Language Understanding

Once the system has a set of hypotheses for what the user said, it must try to understand what was meant by these words. To a dialogue system, the exact meaning of the words

is unimportant. What really matters is what the user was trying to achieve by saying the words. For example, whether the user says “I’d like the phone number.” or “What is the phone number?” the desired outcome is the same. The user is asking for the phone number. The fact that the first utterance is a statement and the second is a question is irrelevant. This distinction between the exact semantics of an utterance and its purpose was first made explicit in the definition of a *speech act*, which is a representation of this underlying action (Austin 1962; Searle 1969). In the example above, the speech act for both utterances would be “request”.

The original idea of speech acts has been extended in the case of dialogue to include actions relating to turn-taking, social conventions and grounding (Traum 1999). The resulting concept is called a *dialogue act tag*. Dialogue act tags also allow actions such as “confirm” and “affirm” for confirmations and affirmations. For example, a “confirm” action might be used to represent “Did you say you wanted Chinese food?” and an “affirm” act might be used to represent “Yes!”.

In the traditional definitions of both speech and dialogue acts, the semantic information is completely separated from the act. A simplified form of semantic information is clearly an important input to the dialogue system. In the case of the user asking for the phone number it is important to the system that it is the phone number that is being requested. It therefore makes sense to include this and to represent the dialogue act as “request(phone)”. Similarly, in the confirmation case above it is more appropriate to represent the act as “confirm(food=Chinese)”. In this thesis, the traditional concept of a dialogue act tag will be called the *dialogue act type* and simplified semantic information is joined with it to give the full dialogue act. Dialogue acts are therefore represented as the combination of the dialogue act type followed by a (possibly empty) sequence of *dialogue act items*,

$$\text{acttype}(\underbrace{a = x, b = y, \dots}_{\text{act items}}).$$

The *acttype* denotes the type of dialogue act while the act items,  $a=x$ ,  $b=y$ , etc., will be either attribute-value pairs such as  $\text{type}=\text{Chinese}$  or simply an attribute name or value. Examples of the latter cases include  $\text{request}(\text{addr})$ , meaning “What is the address?” and  $\text{inform}(=\text{dontcare})$ , meaning “I don’t care”. A detailed description of the dialogue act set used in this thesis is given in Appendix A.

With the concept of dialogue acts in hand, the task of understanding the user becomes one of deciphering dialogue acts. This is known as *semantic decoding*. In general one could imagine doing this on the basis of several sensory inputs. The pitch of a user’s utterance might, for example, give some indication as to the dialogue act type. Of course, the simplest approach is to simply use the output of the speech recogniser, and that is the method that is typically used.

There are a wide range of techniques available for semantic decoding. Template matching and grammar based methods are two examples of hand-crafted techniques. Data-driven approaches include the Hidden Vector State model (He and Young 2006), machine translation techniques (Wong and Mooney 2007), Combinatory Categorical

Grammars (Zettlemoyer and Collins 2007), Support Vector Machines (Mairesse et al. 2009) and Weighted Finite State Transducers (Jurcicek et al. 2009).

Most semantic decoders will assign exactly one dialogue act for each possible word sequence obtained from the speech recogniser. In the case of ambiguities, however, the semantic decoder may choose to output a list of the most probable outputs along with associated confidence scores. Since the speech recogniser is producing an  $N$ -best list of word sequences, some method must be found for combining the confidence scores from the speech recogniser with those of the semantic decoder (see Appendix F for more details).

### 2.1.3 Decision Making

Once a set of possible dialogue acts has been decoded, the system must choose an appropriate response. The component which makes these decisions is called the *dialogue manager*. The response chosen by the system is encoded as a dialogue act and is called the *system action* or *system act*. The topic of dialogue management will be the major focus of this thesis.

The chosen response is selected from a set of possible actions,  $a \in \mathcal{A}$  and will depend on the input that the system receives from the semantic decoder. This input is called the *observation*, labelled  $o \in \mathcal{O}$ , since it encodes everything that the system observes about the user.

Choosing the best action requires more knowledge than simply the last observation. The full dialogue history and context also play an important role. The dialogue manager takes this into consideration by maintaining an internal representation of the full observation sequence. This is called the *dialogue state*, *system state* or *belief state*,<sup>1</sup> and is denoted by  $b \in \mathcal{B}$ . The current belief state will depend on a *belief state transition function* which takes a given belief state and updates it for each new observation and system action. This transition function is therefore a mapping  $T : \mathcal{B} \times \mathcal{A} \times \mathcal{O} \longrightarrow \mathcal{B}$ .

The component of the dialogue manager which defines its behaviour is the *dialogue policy*,  $\pi$ . The policy determines what the system should do in each belief state. In general, the policy will define a probability distribution over which actions might be taken. If  $\Pi(\mathcal{A})$  denotes the set of these distributions then the dialogue policy will be a mapping from belief states to this set,  $\pi : \mathcal{B} \longrightarrow \Pi(\mathcal{A})$ .

Clearly the actions, belief states and observations are all indexed by the turn number. When it is important to note the time step being considered, they are denoted  $a_t$ ,  $b_t$  and  $o_t$ . While the system is in state  $b_t$  it will choose action  $a_t$  according to the distribution determined by the policy,  $\pi(b_t)$ . The system then observes observation

---

<sup>1</sup> The name *belief state* is traditionally reserved in the literature for systems that use a particular statistical assumption, called “partial observability” (Sect. 2.3.2). However, even when this model is not used, the system’s internal state will always be a representation of its beliefs about what has happened in the dialogue. It is therefore reasonable to use the term for all models.

$o_{t+1}$  and transitions to a new system belief state  $b_{t+1}$ . When exact point in time is insignificant, the  $t$  is omitted and a prime symbol is used to denote the next time step (e.g.  $o' = o_{t+1}$ ).

### 2.1.4 Response Generation

The final stage of the dialogue cycle is to convey the system's response to the user. This is done in two steps. First a *natural language generator* converts the system dialogue act to text. The output is then passed to a *text-to-speech (TTS) engine* which conveys the message as audio.

The simplest approach for natural language generation is to use templates. As an example, a template might transform “inform(phone= $x$ )” into “The phone number is  $x$ ”, where “ $x$ ” may be replaced by any phone number. Templates have proven to be relatively effective for natural language generation, since the number of system dialogue acts is reasonably tractable. More complex approaches have also been developed, as for example in Mairesse and Walker (2007).

In the area of text-to-speech there are many alternative approaches. One of the most popular is the *unit selection* approach, which splices segments of speech from a database to generate sound for a given word sequence. Example systems which typically take this approach are Festival (Clark et al. 2004) and FLite (Black and Lenzo 2001). An alternative approach is to use a Hidden Markov Model (HMM) to generate the required audio, as is done in the HTS system (Zen et al. 2007).

### 2.1.5 Extensions to the Dialogue Cycle

The dialogue cycle, as it is shown in Fig. 2.1, is slightly restrictive for more general human-computer interactions. The figure doesn't allow for any method of including information other than speech. Fortunately, non-speech inputs can easily be included by adding inputs to and outputs from the dialogue manager. This does not break the dialogue cycle structure since the dialogue manager can extend its definition of state to include this extra information. Similarly, the output from the dialogue manager can be extended to include more than a dialogue act and could include other actions as well. An example of such a system is given in Williams (2007a), which uses non-speech actions and inputs to help troubleshoot an internet connection. Bohus and Horvitz (2009) provides an example of a spoken dialogue system with multiple parties and visual sensory inputs.

The other restriction imposed by the dialogue cycle is the structure of the turns. In real systems it is not always obvious when the turns start and end, and real-users will often speak out of turn. This is completely natural to humans, who will often say things like “Yes”, “Uh-huh”, etc. to indicate they are listening, sometimes finish other people's sentences and refuse to behave in a turn-taking manner. Some attempt

at breaking away from turn-based systems is given in Rudnicky and Xu (1999), although many systems still typically employ the turn-based architecture.

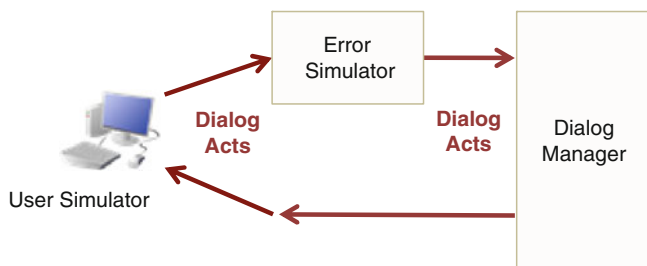
## 2.2 User Simulation

Testing a dialogue manager with human users can be a very time consuming task. Simulated environments provide one way of speeding up the development process by providing a more efficient testing mechanism. A simulated environment will often generate situations that the dialogue system designer will not have thought about and the system can be refined to handle them. A recent area of research interest is to build dialogue managers that can learn automatically what actions to take. In the case of these systems, a simulated environment is particularly important as the system can be boot-strapped by learning from interactions with the simulator. Further refinements can then be obtained from real interactions with human users.

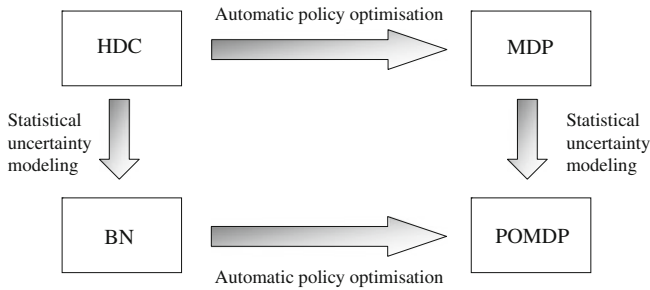
Figure 2.2 shows a graphical representation of the adjusted dialogue cycle when using a simulator. The figure shows the standard approach to simulation, which is to operate at a dialogue act level. A *user simulator* generates dialogue acts given the past dialogue history, as if it were human. This is passed through an *error simulator* which generates appropriate confusions and confidence scores.

There are also simulation environments which have been built to operate at a word-level (Schatzmann et al. 2007a; Jung et al. 2009). In this case, the simulated dialogue act is used to generate a word-level form, which is passed to the error-simulator to produce word-level confusions. This is then passed to the semantic decoding component as in the human-machine case.

A wide range of data-driven simulation techniques are available. Some examples are bigram models (Levin and Pieraccini 1997), goal-based models (Pietquin and Renals 2002; Scheffler and Young 2001), conditional random fields (Jung et al. 2009) and hidden agenda models (Schatzmann et al. 2007b). A survey of user simulation approaches is given in Schatzmann et al. (2006).



**Fig. 2.2** A graphical representation of the dialogue cycle with a user simulator instead of a human user



**Fig. 2.3** A comparison of four major paradigms in spoken dialogue management. The figure contrasts hand-crafted approaches (HDC), Markov Decision Process models (MDP), Bayesian Network approaches (BN) and Partially Observable MDP approaches (POMDP) with respect to how they handle uncertainty and policy optimisation

## 2.3 The Dialogue Manager

The above sections have introduced each of the components of a dialogue system. The rest of the thesis will assume that components for speech recognition, spoken language understanding, response generation, user simulation and text-to-speech are all given, and will focus on how to build an effective dialogue manager.

Two of the most important questions concerning a dialogue manager are:

1. Is uncertainty handled with a statistical approach?
2. Are policies optimised automatically?

The way that these two questions are answered gives rise to four different paradigms for building the dialogue manager. Figure 2.3 shows how the four paradigms contrast with one another. A literature review of research in each of the four approaches is presented below, with particular emphasis on systems which use a statistical approach to handle uncertainty and optimise policies automatically.

### 2.3.1 Hand-Crafted Dialogue Management

At the most basic level, the concepts of belief state, state transitions and policy are all that are needed to define a dialogue manager. Many dialogue management frameworks allow the system designer to directly define all of these components. Because they are directly defined, the components are said to be *hand-crafted*. As dialogues become more complex, the number of states, transitions, and policy decisions becomes very large so researchers have developed various techniques to facilitate the design process.

One of the simplest approaches is to represent the dialogue manager as a graph or flow-chart, sometimes called the *call-flow* (Sutton et al. 1996; McTear 1998; Pieraccini and Huerta 2008). Nodes in the graph represent belief states of the dialogue

and define which action should be taken, while transitions are determined by the observations received. This approach has proven to be very effective when the system's prompts elicit highly restricted responses from the user. On the other hand, the call-flow model typically struggles when users take the initiative and direct the dialogue themselves.

An alternative approach is the idea of a *frame-based dialogue manager*, also known as a *form-filling dialogue manager* (Goddeau et al. 1996). Frame-based approaches assume a set of concepts that the user can talk about, called *slots*, which take on values from a pre-defined set. The current set of filled slots is included as part of the state, along with some hand-crafted information about how certain the system is about each slot value. Dialogue management proceeds by using a pre-specified action for each set of known slots. The ability for users to speak about any slot at any time allows for much more freedom in the dialogue, which is typically perceived as more natural for users.

The frame-based approach is most often used in *information-seeking dialogues*, where a user is seeking information subject to a set of constraints. One example is in the case of a tourist information system, which might have slots for the price range, type of venue, and area. The system would ask for slot-values until it decided that enough have been given, at which point it would offer information about a relevant venue.

A problem with frame-based approaches is that some of the slots will not be relevant for particular dialogues. Another issue is that dialogues are often composed of smaller sub-dialogues, and the progress in a dialogue often follows a particular agenda, which cannot easily be modelled using frames. This has led to the development of *hierarchical* and *agenda-based dialogue managers* (Rudnicky and Xu 1999; Bohus and Rudnicky 2003). These approaches allow the definition of sub-dialogues which depend on the values of slots at higher levels.

When the domain of discourse moves away from information-seeking tasks, then the agenda-based and frame-based approaches sometimes struggle to adapt. Some researchers have suggested that instead of trying to determine only what the user wants, one should aim to determine a shared plan. This is particularly relevant in collaborative dialogues, such as language learning, where both participants must work together to achieve a task. Dialogue systems based on this idea are said to use *plan-based dialogue managers* (Rich et al. 1998).

Another popular framework for dialogue management is the *information state model* (Larsson and Traum 2001; Bos et al. 2003). A central idea in this framework is that dialogue acts correspond to dialogue moves and are used to update an information state subject to certain preconditions. The information state represents the accumulation of everything that has happened in the dialogue, and is used by the dialogue manager to choose its next action according to an update strategy. Logic programming is one possible method of implementing this update strategy (Fodor and Huerta 2006).

All of the above approaches give important insights into how to structure a dialogue manager. However, none of the approaches provide a principled approach to handling the uncertainty in dialogue or to deciding which actions to take at a



particular point. This has led to research into frameworks which do include principled approaches for each of these issues.

### 2.3.2 Partial Observability

Uncertainty is a key issue that must be dealt with by any spoken dialogue system. Both speech recognition and semantic decoding have high error rates and this must be taken into account by the dialogue manager. The traditional approach to handling this uncertainty is simply to augment the hand-crafted belief state with states representing the uncertainty (Bohus and Rudnicky 2005). This has the advantage that it fits naturally into the frameworks presented above. The disadvantage is that it is not necessarily clear how to provide a principled definition for these states of uncertainty.

The alternative is to build a model of the uncertainty directly into the system's belief state. The user's goals and other aspects of the environment are considered as partially observable random variables which must be inferred from the observations. The probability distribution over these random variables give a well-founded representation of the uncertainty, and may be calculated using Bayesian Networks (Pulman 1996) or other techniques (see Chap. 3 for an introduction to Bayesian Networks). Implementations of this idea have been given by Horvitz and Paek (1999) and Meng et al. (2003). In both cases, policy decisions are made using hand-crafted rules. Meng et al. (2003) use thresholds to decide whether a value should be accepted or not while the Value of information is used to decide on the next action in Horvitz and Paek (1999).

One formal model of this idea is to define a set of possible environment states,  $s \in \mathcal{S}$ , along with an observation probability function,  $p(o|s)$ . Note that the observations are assumed to depend conditionally on only the environment state. The change in the environment states is assumed to follow a *stationary distribution* (it does not change with time), where the probability of a given environment state,  $s_{t+1} = s'$ , depends only on the previous state,  $s_t = s$ , and the last machine action,  $a_t = a$ . This dependency is governed by the probability function  $p(s'|s, o)$ . The assumption that environment states depend only on their previous value is sometimes called the *Markov assumption*.

Under the above assumptions, a statistically-motivated approach to handling the uncertainty may be found. Denoting the state at time  $t$  by  $s_t$ , then Bayes' theorem shows that when a new observation,  $o_{t+1} = o'$ , is received then:

$$p(s_{t+1} = s') \propto \sum_{s \in \mathcal{S}} p(s_t = s) p(s_{t+1} = s' | s_t = s, a_t = a) p(o_{t+1} = o' | s_{t+1} = s'). \quad (2.1)$$

Now the system's belief state at each point in time,  $b_t$ , is defined as the probability distribution over states given all observations up to that point. In particular,  $b_t(s)$  denotes the probability of state  $s$  at time  $t$  given all observations up to time  $t$ . The set of all possible belief states is the set of probability distributions over environment

states,  $\mathcal{B} = \Pi(\mathcal{S})$ . Rewriting Eq. 2.1 in terms of the new belief state  $b'$  and old belief state  $b$  gives:

$$b'(s') \propto \sum_{s \in \mathcal{S}} b(s) p(s'|s, a) p(o'|s'). \quad (2.2)$$

This update of the belief state gives a formula defining the belief state transition function,  $T$ , which determines the transitions of the dialogue manager's internal state. In practice the set of environment states will be very large, since the environment state must represent all information that is necessary for making decisions about what to say. To do this, the environment state will need to incorporate both the user's goal and the dialogue history. In a system with a number of slots, every different combination of the slot-values will constitute one possible user goal. The number of user goals therefore grows exponentially with the number of slots, which causes a serious computational burden.

Chapter 3 will discuss methods for computing this belief update efficiently. The use of this statistically motivated framework has the potential to significantly improve system performance, as will be shown in Chap. 6.

### 2.3.3 Policy Learning

Another key issue in spoken dialogue management is the design of the dialogue policy. While it is possible to design this by hand, such an approach has several shortcomings. Development is manually intensive and expensive, the systems are difficult to maintain and performance is often sub-optimal.

An alternative to hand-crafting policies is to define an objective measure of performance and to optimise this automatically. A reward is assigned to each turn which depends on the belief state,  $b$ , and action taken,  $a$ , and is denoted by  $r(b, a)$ . The measure of performance in a dialogue is then the sum of turn-based rewards,

$$R = \sum_{t=1}^T r(b_t, a_t), \quad (2.3)$$

where  $T$  is the number of dialogue turns.

The progression of a dialogue cannot be determined before one sees how the user responds to the actions taken by the system. Because of the uncertainty in how the user will behave, the reward under a given system policy will be stochastic. A sensible objective for the system is therefore to optimise the expected or mean value of this reward,  $\mathbb{E}(R)$ . Systems that attempt to optimise this reward are said to use *reinforcement learning* because good policies are *reinforced* by good rewards.

In order to compute this optimisation, further assumptions must be made. The most common approach is to assume that changes in the belief state are *Markov* (i.e. the probability of a new belief state  $b'$  depends only on the previous belief state and system action, and not the full history). This model is known as the *Markov Decision*

*Process (MDP)* (Sutton and Barto 1998). Chapter 5 will discuss optimisation methods for this model in detail.

Traditional work on the MDP model assumes that the belief space,  $\mathcal{B}$ , is finite. This makes policy optimisation significantly simpler, but is not a necessary assumption. This thesis will also consider the more general case of continuous belief state MDPs.

The Markov Decision Process was first suggested as a dialogue model by Levin and Pieraccini (1997). Their original system was trained using a bigram model for user simulation and was able to optimise for a particular reward using standard algorithms (Levin et al. 2000).

When one moves to working with human users, the rewards are difficult to define as the system does not actually know what the user wants. What can be found is an estimate of the user satisfaction, by using regression on known features of the dialogue, as suggested in the PARADISE framework (Walker et al. 1997). This approach has been used successfully to build an MDP based system for real users (Walker 2000). There are also various other systems which have been tested with human users (Singh et al. 2002; Denecke et al. 2005).

Significant research in the area of automatic policy learning has been dedicated to increasing the complexity of the available dialogues, which depends largely on the learning algorithm. Various algorithms have been tested, including Value Iteration (Singh et al. 2002), Q-learning (Schatzmann et al. 2005), Monte-Carlo learning (Pietquin and Renals 2002),  $Q(\lambda)$  learning (Scheffler 2002) and SARSA( $\lambda$ ) learning with linear function approximation (Henderson et al. 2005). These systems were all trained using a simulated user.

With a traditional MDP, the state transitions are still hand-crafted, and must be defined by the system designer. As with fully hand-crafted systems, the state set can become difficult to define when the dialogues become more complex. The ideas used by the fully hand-crafted systems transfer naturally to the MDP case and can be employed without much alteration. For example, information state updates have been used in MDP systems by simply adding the definition of rewards and the Markov assumption to the standard information state model (Lemon et al. 2006; Heeman 2007).

The central assumption in the above models is that the system's belief state depends only on its previous value and the last system action—the Markov assumption. There has been some work showing that this may not necessarily be a valid assumption (Paek and Chickering 2006). More general influence diagrams (Paek and Chickering 2006) and semi-Markov decision processes (Cuayhuil et al. 2007) have been suggested as alternative approaches to the reinforcement learning of dialogue managers which overcome this issue.

### 2.3.4 Partially Observable Markov Decision Processes

The idea of combining the statistical approach to learning and to handling uncertainty results in a model known as the *Partially Observable Markov Decision Process*

(*POMDP*), which was first suggested for dialogue by Roy et al. (2000). The model makes the same assumptions as all partially observable models and therefore includes the concepts of: an environment state,  $s$ , the Markov property on the environment state, environment state transition probabilities and observation probabilities.

The assumptions made to enable learning are slightly different. Instead of defining rewards as a function of belief states, as is done in the MDP model, the reward is defined as a function of the environment state,  $s$ . This change corresponds to the intuitive idea that performance should be measured as a function of what the user *really* wants and not simply what the system believes. The aim of the system is then to optimise the expected dialogue reward using this adjusted reward function.

Theoretical examination has shown that this adjusted approach in POMDPs is not really significantly different from the standard MDP. The Markov assumption on the environment state will always result in the Markov assumption being true on the belief state (Kaelbling et al. 1998), so there is no need to include this as an extra assumption. Although the rewards are defined as a function of the true environment state rather than the belief state, a belief state reward may be obtained from this using the formula,

$$r(b, a) = \sum_{s \in \mathcal{S}} b(s)r(s, a) \quad (2.4)$$

The POMDP assumptions therefore always result in an MDP, and to a large extent one can restrict the work on policy learning to the MDP model. An important feature of POMDPs is that the belief states represent probabilities and are always continuous. This representation as an MDP only applies if one allows the MDP to have a continuous belief state space. Another important assumption that is almost always made with POMDP models is that the observation space is finite. This assumption does not affect the assumptions of the MDP model and will not be required here.

Although the POMDP model provides an elegant approach to handling both uncertainty and policy optimisation, it results in significant computational complexity. Roy et al. (2000)'s original model allowed for only 13 environment states, 20 actions and 16 observations. Research in POMDP models for dialogue has been focused largely on reducing the computational burden, so that the systems may engage in more complex dialogues.

There are two key areas which are to blame for this computational complexity. Firstly, the belief updating equation (Eq. 2.2 of Sect. 2.3.2) requires a computation that is quadratic in the number of environment states. In even the simplest of dialogues, this number is unmanageable as it will grow exponentially with the number of slots. Secondly, the use of a probabilistic approach to modelling the uncertainty means that the belief state space is continuous. The standard algorithms used for policy learning do not extend to this case, and so special learning algorithms must be used. The algorithms that have been developed for exact POMDP optimisation have thus far been unable to extend past a very small number of environment states.

A third key area which is important for building a POMDP-based dialogue manager is the design of the model parameters. The observation and transition probabilities have a large potential effect on the progression of the system's belief state and

must be appropriately chosen. A discussion of current approaches to this issue, as well as to the belief updating and policy optimisation issues is given below. This is followed by a survey of past POMDP-based dialogue managers.

#### 2.3.4.1 POMDP Model Parameters

The simplest approach to choosing model parameters is to allow the system designer to specify them directly. The parameters can be tuned by testing the system in both simulated and real-world environments. While this approach is not particularly well-founded it has been used in most POMDP systems because of the complexity of alternative approaches, and results have been successful (Roy et al. 2000; Zhang et al. 2001; Young et al. 2009; Thomson et al. 2008; Bui et al. 2009).

Ideally, one would like to use a corpus to estimate the model parameters from real interactions. Data-driven parameters may be obtained by annotating the correct dialogue state and using maximum-likelihood estimates. This approach has been used but has been restricted to cases where the user's goal remains constant and is simple to annotate, as in Williams (2008b) and Kim et al. (2008).

Estimates could potentially be obtained without annotation of dialogue states, by first considering the state as hidden and using estimated based on the most likely state sequence and using this to estimate the model parameters. This process is called Viterbi estimation and is used in Doshi and Roy (2007). Current research on this approach has been restricted to dialogues with a very small number of environment states [Doshi and Roy (2007) uses 7 environment states].

Chapter 7 provides a way to scale the idea of parameter learning without annotated dialogue states to more complex dialogues. The approach considered there also provides an approximation to the full posterior distribution over parameters instead of only providing maximum likelihood estimates.

#### 2.3.4.2 Belief Updating

A simple method for improving the efficiency of belief updates is to use Dynamic Bayesian Networks as in Zhang et al. (2001); Kim et al. (2008); Thomson et al. (2008) and Bui et al. (2009). This reduces the amount of computation required by exploiting conditional independence assumptions between slots. Computation is further reduced by using only the marginal distribution for the previous time-slice during updates (Boyen and Koller 1998), where the marginal distribution is the distribution of only one variable, obtained by summing out all others. The resulting equations reduce the computational complexity of having large numbers of slots but do not change the effect of an increase in the number of values per slot. With very large numbers of possible values, the resulting updates may still require too much computation for real-time operation.

Particle filters are a second method that has been proposed for reducing belief update computation times (Williams 2007b). The particle filter approach represents

the environment state as a Bayesian Network but then samples from appropriate distributions to compute a Monte-Carlo estimate of the updated belief distribution.

An alternative simplification that can be made is to partition the environment states into groups which are indistinguishable, as proposed in the Hidden Information State framework (Young et al. 2005, 2009). A central requirement of this framework is that the user’s goal remains constant through the dialogue. Under this assumption, updates may be done efficiently because the probability update equation is the same for all the indistinguishable states.

Henderson and Lemon (2008) suggest a fourth approach, which was shown to be very similar to the approach used by the HIS system. The belief state is thought of as a probability distribution over possible MDP states, which are defined using hand-crafted rules. Rather than inferring the goal from a generative model of the observations, the observation probabilities are used to generate the relevant probability for each MDP state. Since each MDP state represents a belief distribution over the environment states, a weighted combination gives the overall POMDP belief state.

A fifth approach to efficiently updating the beliefs will be outlined in Chap. 3. The idea there is to maintain only a  $k$ -best list for each node in a Bayesian Network. This allows the efficiency improvements from using Bayesian Networks to be used in conjunction with the improvements resulting from the Hidden Information State approach. Changes in the goal are allowed and the model has been implemented successfully for the same domain as the Hidden Information State model. Most importantly, this approach scales with both the number of slots in the system as well as the number of slot values.

### 2.3.4.3 POMDP Policy Optimisation

Exact POMDP optimisation algorithms do exist but they are intractable for all but the simplest problems (Kaelbling et al. 1998; Cassandra et al. 1997). In order to usefully apply the POMDP model to dialogue, researchers have been forced to use approximate algorithms for the optimisation. Even approximate algorithms do not generally scale to the size of the problem found in spoken dialogue so methods have also been developed to incorporate domain knowledge into the algorithms.

The concept of the Summary POMDP was one of the first ideas developed for this purpose (Williams and Young 2005). Instead of optimising over all possible actions, the actions are grouped together into *summary actions*. This grouping is implemented by the system designer, who uses domain knowledge to exploit the fact that one of the actions in the group is always better than the others. Similarly the belief state space is mapped into a lower dimensional *summary space* where optimisation can proceed more easily. The original actions and belief states in this framework are called *master actions* and *master states*, and are said to operate in the *master space*. More detail on the idea of summary actions is given in Sect. 5.2.

Even more domain knowledge can be included by augmenting the POMDP belief state with an MDP-style state (Williams 2008a,b). The system designer chooses a

set of available actions for each MDP state, and the POMDP optimisation proceeds using only those actions. This constrains the search space dramatically and can induce large improvements in efficiency. The approach is particularly useful for encoding business rules, where the system designer may want to force a particular sequence of utterances. An example where this might be used is to restrict a phone-based bank system from transferring money without some form of authentication.

In many dialogue systems, the belief state can be factorised according to a set of slots, much like in the frame-based approaches used for traditional systems. Previous researchers have shown how this can be exploited to enable tractable policy learning. For example, Williams and Young (2006, 2007) developed the Composite Summary Point Based Value Iteration algorithm, which optimises the policy for each slot separately. Overall system actions are selected from the slot-level actions using heuristic rules. A similar approach is taken in the Dynamic Decision Network-POMDP (DDN-POMDP) algorithm developed by Bui et al. (2009).

The use of heuristic rules for actions at the overall level is clearly restrictive on the available choices made during policy optimisation. For this reason, methods that can learn decisions at a master-space level may well be preferred. A useful technique for implementing this is to use function approximation. Grid-based approximations have been used in optimising dialogue policies by Young et al. (2009) and Zhang et al. (2001). More general linear approximations for POMDP systems were suggested in Thomson et al. (2008) and the work of Li et al. (2009) extends this idea by enabling the system to automatically select useful features [Note that similar linear approximations had already been used in MDP systems as for example in Henderson et al. (2005)]. Further details of these ideas are discussed in Chap. 5.

#### 2.3.4.4 Survey of POMDP Dialogue Managers

The complexity of previous dialogue managers depends largely on the assumptions made and the learning algorithms used. The original systems of Roy et al. (2000) and Zhang et al. (2001) made very few assumptions but operated with very few environment states. Roy et al. (2000) used the Augmented MDP algorithm for learning and built a system with 13 environment states. The system in Zhang et al. (2001) used Grid-based Q learning to optimise a system with 40 states.

The dialogue system of Doshi and Roy (2007) has only 7 environment states but focuses on handling uncertainty in the parameters rather than allowing for more complex dialogues. The model parameters (observation and transition probabilities) are included as unknown parameters so that the belief state also includes the uncertainty in them. This enables the system to learn a user model during operation, but demands a very limited domain of discourse.

The Composite Summary Point Based Value Iteration algorithm, suggested by Williams and Young (2007), has been tested for a system with 5 slots, and 100 values for each slot. It represented a large increase in complexity compared with previous systems. This scalability was achieved through the summary ideas, heuristics for choosing master-level actions and the assumption of a frame-based system.



Particle filters have been implemented for belief updating in a troubleshooting system with 20 factors, where two factors have 13 and 11 possible values and the remaining factors have either 2 or 3 possible values (Williams 2007b). Belief updating computation times were reduced by a factor of around 10 when compared to exact inference.

Bui et al. (2009) give real-time performance with up to 1000 different slots each having 10 slot values or 1 slot with 500 slot values. In the 1000 slot case, Bui et al. (2009) exploit the fact that observations only affect a limited number of the slots and only these slots need be updated. Policy learning is implemented with an algorithm called the Dynamic Decision Network-POMDP (DDN-POMDP). Tractability is achieved by using heuristics for choosing master-level actions, the assumption of independent slots and a finite length lookahead during action selection.

The Hidden Information State model has been implemented to build a tourist information dialogue manager with 10 slots, with the number of slot-values ranging from 4 to 45 (Young et al. 2005; 2009). Learning has been implemented using grid-based Monte-Carlo learning as well as a  $k$ -best Monte-Carlo algorithm (Lefevre et al. 2009). The major assumptions in the HIS model are that user goals may be grouped into partitions and that user goals remain constant. The mixture POMDP model of Henderson and Lemon (2008) was implemented on the same task and used a Q-MDP learning algorithm.

Another real-world POMDP dialogue system is the flight-booking system built using Heuristic Search Value Iteration by Kim et al. (2008). This system has four slots and ignores the actual slot-values, so that the belief distributions operate over user intentions rather than user goals.

Least Squares Policy Iteration is a relatively new algorithm for dialogue optimisation and has been used to implement a voice dialler with 50000 employee names (Li et al. 2009). The system is also able to disambiguate between employees with the same name by using their location and can choose between different phone numbers for a particular employee (e.g. mobile, home or work). A separate POMDP-based dialogue manager for this task was also developed using value iteration (Williams 2008b).

The algorithm suggested for policy optimisation in this thesis is Natural Actor Critic (Peters et al. 2005; Peters and Schaal 2008). This approach has been used in conjunction with the state updating techniques of Chaps. 3 and 4 to build a real-world tourist information system for the same domain as the Hidden Information State model (Thomson et al. 2008). The approach scales with both the number of slots and the number of slot-values, and policy optimisation is performed in such a way that the system can learn between many different master-level actions.

## References

- Austin JL (1962) How to do things with words. Oxford University Press, New York  
 Black AW, Lenzo KA (2001) FLite: a small fast run-time synthesis engine. In: ISCA tutorial and research workshop on speech, synthesis



- Bohus D, Horvitz E (2009) Models for multiparty engagement in open-world dialog. In: Proceedings of SIGDIAL, pp 225–234
- Bohus D, Rudnicky A (2003) Ravenclaw: dialog management using hierarchical task decomposition and an expectation agenda. In: Proceedings of Eurospeech
- Bohus D, Rudnicky AI (2005) Constructing accurate beliefs in spoken dialog systems. In: Proceedings of ASRU, pp 272–277
- Boys J, Klein E, Lemon O, Oka T (2003) DIPPER: description and formalisation of an information-state update dialogue system architecture. In: Proceedings of SIGDIAL, pp 115–124
- Boyen X, Koller D (1998) Tractable inference for complex stochastic processes. In: Proceedings of uncertainty in AI. Morgan Kaufmann, San Francisco, pp 33–42
- Bui T, Poel M, Nijholt A, Zwiers J (2009) A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. *Nat Lang Eng* 15(2):273–307
- Cassandra A, Littman ML, Zhang NL (1997) Incremental pruning: a simple, fast, exact method for partially observable Markov decision processes. In: Proceedings of uncertainty in AI, pp 54–61
- Clark RA, Richmond K, King S (2004) Festival 2-build your own general purpose unit selection speech synthesiser. In: Proceedings of the ISCA workshop on speech, synthesis
- Cuayhuil H, Renals S, Lemon, Shimodaira OH (2007) Hierarchical dialogue optimization using semi-Markov decision processes. In: Proceedings of interspeech
- Denecke M, Dohsaka K, Nakano M (2005) Fast reinforcement learning of dialogue policies using stable function approximation. In: Proceedings of IJCNLP. Springer, Heidelberg, pp 1–11
- Doshi F, Roy N (2007) Efficient model learning for dialog management. In: Proceedings of the ACM/IEEE international conference on human-robot interaction, pp 65–72. ACM. ISBN 978-1-59593-617-2
- Evermann, G Woodland PC (2000) Large vocabulary decoding and confidence estimation using word posterior probabilities. In: Proceedings of ICASSP
- Fodor P, Huerta JM (2006) Planning and logic programming for dialog management. In: Proceedings of SLT, pp 214–217
- Goddeau D, Meng H, Polifroni J, Seneff S, Busayapongchai S (1996) A form-based dialogue manager for spoken language applications. In: Proceedings of ICSLP
- He Y, Young S (2006) Spoken language understanding using the hidden vector state model. *Speech Commun* 48(3–4):262–275
- Heeman PA (2007) Combining reinforcement learning with information-state update rules. In: Proceedings of HLT/NAACL, pp 268–275
- Henderson J, Lemon O (2008) Mixture model POMDPs for efficient handling of uncertainty in dialogue management. In: Proceedings of ACL/HLT, pp 73–76. Association for Computational Linguistics
- Henderson J, Lemon O, Georgila K (2005) Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In: IJCAI workshop on knowledge and reasoning in practical dialogue systems, pp 68–75
- Horvitz E, Paek T (1999) A computational architecture for conversation. In: Proceedings of the seventh international conference on user modeling. Springer, Wien, pp 201–210
- Jiang H (2005) Confidence measures for speech recognition: a survey. *Speech Commun* 45(4):455–470
- Jung S, Lee C, Kim K, Jeong M, Lee GG (2009) Data-driven user simulation for automated evaluation of spoken dialog systems. *Comput Speech Lang* 23(4):479–509
- Jurcicek F, Gasic M, Keizer S, Mairesse F, Thomson B, Yu K, Young S (2009) Transformation-based learning for semantic parsing. In: Proceedings of SIGDIAL
- Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artif Intell* 101(1–2):99–134
- Kim D, Sim HS, Kim KE, Kim JH, Kim H, Sung JW (2008) Effects of user modeling on POMDP-based dialogue systems. In: Proceedings of interspeech
- Larsson S, Traum DR (2001) Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Nat Lang Eng* 6(3&4):323–340

- Lefevre F, Gasic M, Jurcicek F, Keizer S, Mairesse F, Thomson B, Yu K, Young S (2009) k-nearest neighbor Monte-Carlo control algorithm for POMDP-based dialogue systems. In: Proceedings of SIGDIAL
- Lemon O, Georgila K, Henderson J, Stuttle M (2006) An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In: Proceedings of EACL
- Levin E, Pieraccini R, Eckert W (2000) A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans Speech Audio Process* 8(1):11–23
- Levin E, Pieraccini R (1997) A stochastic model of computer-human interaction for learning dialogue strategies. In: Proceedings of EUROSPEECH, pp 1883–1886
- Litman DJ, Ros CP, Forbes-Riley K, VanLehn K, Bhembé D, Silliman S (2006) Spoken versus typed human and computer dialogue tutoring. *Int J Artif Intell Educ* 16(2):145–170
- Li L, Williams JD, Balakrishnan S (2009) Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In: Proceedings of interspeech
- Mairesse F, Gasic M, Keizer FJ, Thomson B, Yu K, Young S (2009) Spoken language understanding from unaligned data using discriminative classification models. In: Proceedings of ICASSP
- Mairesse F, Walker M (2007) PERSONAGE: personality generation for dialogue. In: Proceedings of ACL
- McTear MF (1998) Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. In: Proceedings of ICSLP
- Meng H, Wai C, Pieraccini R (2003) The use of belief networks for mixed-initiative dialog modeling. *IEEE Trans Speech Audio Process* 11(6):757–773
- Paek T, Chickering D (2006) Evaluating the Markov assumption in Markov decision processes for spoken dialogue management. *Lang Res Eval* 40(1):47–66
- Peters J, Schaal S (2008) Natural actor-critic. *Neurocomputing* 71:1180–1190
- Peters J, Vijayakumar S, Schaal S (2005) Natural actor-critic. In: Proceedings of ECML. Springer, Heidelberg, pp 280–291
- Pieraccini R, Huerta JM (2008) Where do we go from here? In: Dybkj L, Minker W (eds) Recent trends in discourse and dialogue. Text, speech and language technology, vol 39. Springer, Heidelberg
- Pietquin O, Renals S (2002) ASR system modeling for automatic evaluation and optimization of dialogue systems. In: Proceedings of ICASSP, pp 46–49
- Pulman S (1996) Conversational games, belief revision and Bayesian networks. In: Proceedings of the 7th computational linguistics in the Netherlands meeting
- Raux A, Bohus D, Langner B, Black AW, Eskenazi M (2006) Doing research on a deployed spoken dialogue system: one year of Let's Go! experience. In: Proceedings of ICSLP
- Rich C, Sidner AL, Rich C, Sidner CL (1998) COLLAGEN: a collaboration manager for software interface agents. *User Model User-Adapt Interact* 8:315–350
- Roy N, Pineau J, Thrun S (2000) Spoken dialogue management using probabilistic reasoning. In: Proceedings of ACL
- Rudnicki A, Xu W (1999) An agenda-based dialog management architecture for spoken language systems. In: Proceedings of ASRU
- Schatzmann J, Weillhammer K, Stuttle M, Young S (2006) A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl Eng Rev* 21(02):97–126
- Schatzmann J, Stuttle MN, Weillhammer K, Young S (2005) Effects of the user model on simulation-based learning of dialogue strategies. In: Proceedings of ASRU
- Schatzmann J, Thomson B, Young S (2007a) Error simulation for training statistical dialogue systems. In: Proceedings of ASRU, pp 526–531
- Schatzmann J, Thomson B, Young S (2007b) Statistical user simulation with a hidden agenda. In: Proceedings of SIGDIAL, pp 273–282
- Scheffler K (2007) Automatic design of spoken dialogue systems. Ph.D. thesis, University of Cambridge

- Scheffler K, Young S (2001) Corpus-based dialogue simulation for automatic strategy learning and evaluation. In: *Proceedings of NAACL*
- Searle JR (1969) *Speech acts: an essay in the philosophy of language*. Cambridge University Press, Cambridge
- Singh S, Litman D, Kearns M, Walker M (2002) Optimizing dialogue management with reinforcement learning: experiments with the NJFun system. *J Artif Intell Res* 16(1):105–133
- Sutton R, Barto A (1998) *Reinforcement Learning: an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge
- Sutton S, Novick DG, Cole R, Vermeulen P, de Villiers J, Schalkwyk J, Fenty M (1996) Building 10,000 spoken dialogue systems. In: *Proceedings of ICSLP*
- Thomson B, Schatzmann J, Young S (2008) Bayesian update of dialogue state for robust dialogue systems. In: *Proceedings of ICASSP*, pp 4937–4940
- Traum DR (1999) *Speech acts for dialogue agents*. In: Wooldridge M, Rao A (eds) *Foundation and theories of rational agents*. Kluwer Academic, New York, pp 169–201
- Walker MA, Litman DJ, Kamm CA, Abella A (1997) PARADISE: a framework for evaluating spoken dialogue agents. In: *Proceedings of ACL-EACL*, pp 271–280
- Walker MA (2000) An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *J Artif Intell Res* 12:387–416
- Walker W, Lamere P, Kwok P, Raj B, Singh R, Gouvea R, Wolf P, Woelfel J (2004) Sphinx-4: a flexible open source framework for speech recognition. Technical Report TR-2004-139, Sun Microsystems
- Williams JD (2007a) Applying POMDPs to dialog systems in the troubleshooting domain. In: *Proceedings of the HLT/NAACL workshop on bridging the gap: academic and industrial research in dialog technology*
- Williams JD (2007b) Using particle filters to track dialogue state. In: *Proceedings of ASRU*
- Williams JD (2008a) The best of both worlds: unifying conventional dialog systems and POMDPs. In: *Proceedings of interspeech*
- Williams JD (2008b) Integrating expert knowledge into POMDP optimization for spoken dialog systems. In: *Proceedings of the AAAI workshop on advancements in POMDP solvers*
- Williams JD, Young S (2005) Scaling up POMDPs for dialog management: the “Summary POMDP” method. In: *Proceedings of ASRU*, pp 177–182
- Williams JD, Young S (2006) Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI). In: *Proceedings of the AAAI workshop on statistical and empirical approaches for spoken dialogue systems*
- Williams JD, Young S (2007) Scaling POMDPs for spoken dialog management. *IEEE Trans Audio Speech Lang Process* 15:2116–2129
- Wong YW, Mooney R (2007) Learning synchronous grammars for semantic parsing with lambda calculus. In: *Proceedings of ACL*, pp 960–967
- Young SJ, Williams JD, Schatzmann J, Stuttle MN, Weilhammer K (2005) The hidden information state approach to dialogue management. Technical Report CUED/FINFENG/TR.544, Cambridge University Engineering Department
- Young S, Kershaw D, Odell J, Ollason D, Valtchev V, Woodland P (2000) *The HTK book version 3.0*. Cambridge University Press, Cambridge
- Young S, Gasic M, Keizer S, Mairesse F, Schatzmann J, Thomson B, Yu K (2009) The hidden information state model: a practical framework for POMDP-based spoken dialogue management. *Comput Speech Lang* 24:150–174. ISSN 08852308
- Zen H, Nose T, Yamagishi J, Sako S, Masuko T, Black AW, Tokuda K (2007) The HMM-based speech synthesis system (HTS) version 2.0. In: *Proceedings of ISCA*, pp 294–299
- Zettlemoyer L, Collins M (2007) Online learning of relaxed CCG grammars for parsing to logical form. In: *Proceedings of EMNLP*, pp 678–687
- Zhang B, Cai Q, Mao J, Chang E, Guo B (2001) Spoken dialogue management as planning and acting under uncertainty. In: *Seventh European conference on speech communication and technology*



<http://www.springer.com/978-1-4471-4922-4>

Statistical Methods for Spoken Dialogue Management

Thomson, B.

2013, XVIII, 138 p., Hardcover

ISBN: 978-1-4471-4922-4