Chapter 10
Dialog Management

Abstract One of the core aspects in the development of conversational interfaces is to design the dialog management strategy. The dialog management strategy defines the system's conversational behaviors in response to user utterances and environmental states. The design of this strategy is usually carried out in industry by handcrafting dialog strategies that are tightly coupled to the application domain in order to optimize the behavior of the conversational interface in that context. More recently, the research community has proposed ways of automating the design of dialog strategies by using statistical models trained with real conversations. This chapter describes the main challenges and tasks in dialog management. We also analyze the main approaches that have been proposed for developing dialog managers and the most important methodologies and standards that can be used for the practical implementation of this important component of a conversational interface.

10.1 Introduction

This chapter describes the main aspects, tasks, and approaches involved in the dialog management (DM) process. Section 10.2 defines the DM process and the tasks involved. To illustrate the complexity of dialog strategy design, this section analyzes two frequently arising design issues: the interaction strategy and the choice of a confirmation strategy.

Dialog management can be classified into handcrafted approaches using rules, which are described in Sect. 10.3, and statistical approaches using machine learning methodologies, which are described in Sect. 10.4. Statistical approaches have been proposed to model the variability in user behaviors and to allow the exploration of a wider range of strategies. This section provides two detailed examples of the practical application of reinforcement learning and corpus-based supervised learning for the development of statistical dialog managers.

M. McTear et al., *The Conversational Interface*,
DOI 10.1007/978-3-319-32967-3_10

10.2 Defining the Dialog Management Task

As has been described in previous chapters, different modules and processes must cooperate to achieve the main goal of a conversational interface. Automatic speech recognition (ASR) is the process of obtaining the text string corresponding to an acoustic input (see Chap. 5). Once the speech recognition component has recognized what the user uttered, it is necessary to understand what was said. Spoken language understanding (SLU) is the process of obtaining a semantic interpretation of a text string (see Chap. 8). This generally involves morphological, lexical, syntactical, semantic, discourse, and pragmatic knowledge.

DM relies on the fundamental task of deciding what action or response a system should take in response to the user's input. There is no universally agreed definition of the tasks that this component has to carry out to make this decision. Traum and Larsson (2003) state that DM involves four main tasks:

1. Updating the dialog context.
2. Providing a context for interpretation.
3. Coordinating other modules.
4. Deciding the information to convey and when to do it.

Thus, the dialog manager has to deal with different sources of information such as the SLU results, results of database queries, application domain knowledge, and knowledge about the users and the previous dialog history. The complexity of DM depends on the task, the extent to which the dialog is flexible, and who has the initiative in the dialog, the system, the user, or both.

Although DM is only one part of the information flow of a conversational interface, it can be seen as one of the most important tasks given that this component encapsulates the logic of the speech application. The selection of a particular action depends on multiple factors, such as the output of ASR (e.g., measures that define the reliability of the recognized information), the dialog interaction (e.g., the number of repairs carried out so far), the application domain (e.g., guidelines for customer service), and the responses and status of external back-ends, devices, and data repositories. Given that the actions of the system directly impact users, the dialog manager is largely responsible for user satisfaction. Because of these factors, the design of an appropriate DM strategy is at the core of conversational interface engineering.

ASR is not perfect, so one of the most critical aspects of the design of the dialog manager involves error handling. The ASR and SLU components make errors, and so conversational interfaces are generally less accurate than humans. For all of this technology to work, severe limitations need to be imposed on the scope of the applications and this requires a great amount of manual work for designers. One common way to alleviate errors is to use techniques aimed at establishing a confidence level for the ASR result and to use that to decide when to ask the user for confirmation, or whether to reject the hypothesis completely and reprompt the user. Too many confirmations as well as too many reprompts could annoy users. So it is

important to reduce the number of confirmations and rejections to a minimum while at the same time preserving a reasonable level of accuracy.

In order to complete the tasks described above and to decide "what to say" and "what to do," the dialog manager needs to track the dialog history and update its representation of the current state of the dialog. In addition, the dialog manager needs a dialog strategy that defines the conversational behavior of the system, for example, when to take the initiative in a dialog or when to establish common ground. The performance of a conversational interface is highly dependent on the quality of its DM strategy.

Unfortunately, the design of a good strategy is far from trivial since there is no clear agreement as to what constitutes a good strategy. User populations are often diverse, thus making it difficult to foresee which form of system behavior will lead to a quick and successful completion of the dialog. In addition, the omnipresence of ASR errors leads to constant uncertainty about the true intentions of the user. As applications become more complex, the process of designing robust dialog strategies becomes very time-consuming and expensive.

To further illustrate the complexity involved in dialog strategy design, the following subsections briefly discuss two frequently arising design issues: interaction strategies and the choice of a confirmation strategy.

10.2.1 Interaction Strategies

The interaction strategy of a conversational interface determines who takes the initiative in the dialog—the system, the user, or both. Three types of interaction strategies are often distinguished in the literature: user-directed, system-directed, and mixed-initiative.

When user-directed initiative is used, the user always has the initiative in the dialog, and the system just responds to the user's queries and commands. The main problem with this strategy is that the user may think that they are free to say whatever they want, which tends to cause ASR and SLU errors. The following example shows a user-directed dialog with the EDECAN dialog system for booking sports facilities at the Technical University of Valencia, Spain (Griol et al. 2014).

System: Welcome to the sports service. How can I help you?
User: I want to book a basketball court.
System: You can see the available courts on the screen.
User: I want a court for November 12th in the afternoon.
System: Sorry, do you want a court from 15 to 21 hours?
User: Yes.
System: There are several courts available on Friday November 12th. Please select one of them.
User: The one at 7 pm.

System: There is only one court with the required features. Do you want to book it?
User: Yes.
System: The basketball court has been booked. Anything else?
User: Yes, I want to know my list of court bookings.
System: You can see the courts on the screen. Anything else?
User: Yes, cancel the one at 12 am.
System: The specified court has been released. Anything else?
User: No, thank you.
System: Thank you for using the sports service. Goodbye.

When system-directed initiative is used, the system has the initiative in the dialog, and the user just answers its queries. The advantage of this strategy is that it helps to constrain the user's input, leading to more efficient dialogs. The disadvantage is a lack of flexibility, as the user is restricted to behaving according to the system's expectations, providing the necessary data to perform some action in the order specified by the system. The following example shows a system-directed dialog with the LUNA dialog system that supports problem-solving tasks in the domain of software/hardware repair (Griol et al. 2014).

System: Help Desk, I am Paola! How may I help you?
User: Hello I have a problem with the printer. It is not printing.
System: Where do you work?
User: For CSI Piemonte.
System: Tell me your name and surname.
User: My name is Ugo Foscolo.
System: I didn't hear you very well. Please repeat and speak louder.
User: Ugo Foscolo.
System: Can you tell me your phone number?
User: Zero eleven forty fifty sixty-three.
System: Can you tell me the brand and model?
User: Epson EPL nine thousand.
System: Is the printer on now?
User: It is on.
System: Can you tell me the message on the printer display?
User: It says No network connection.
System: Can you see the printer from the control panel of your PC?
User: It is active.
System: Now, I will tell you the number of the ticket for your call. Your ticket number is 2609192. Have you noted it down?
User: Yes.

System: Thank you! A technical expert from Help Desk will contact you as soon as possible. Have a nice day!

When the mixed-initiative strategy is used, both the user and the system can take the initiative in the dialog. The advantage is that the system can guide the user in the tasks that are to be performed, while the user can take the initiative, ask questions, introduce new topics, and provide overinformative responses. In the following example of a mixed-initiative dialog with the LUNA dialog system involving the provision of railway information related to timetables, fares, trip times, types of trains, and services the user responds to the system's question about the destination city by stating the day as well as the city (Griol et al. 2008).

System: Welcome to the railway information system. How can I help you?
User: I want to know timetables from Madrid.
System: Do you want to leave from Madrid?
User: Yes.
System: Tell me the destination city.
User: To Granada next Friday.
System: Do you want timetables for next Friday?
User: Yes
System: I have the following information about trains from Madrid to Granada. There is only one train at 8.30. Anything else?
User: No, thank you.
System: Thanks for using this service. Have a good trip.

Although not shown in this dialog, an example of a mixed-initiative dialog in which the user takes the initiative instead of responding directly to the system's question is shown in the following example:

System: Do you want timetables for next Friday?
User: Are there trains before 6 in the morning?

The problem with mixed-initiative dialogs is that the user can potentially say anything and by introducing a different topic may cause the system to lose track of its agenda. Thus, mixed-initiative dialogs require advanced ASR and SLU capabilities as well as the ability to maintain and monitor the dialog history and the system's agenda.

10.2.2 Error Handling and Confirmation Strategies

Because of the limitations of current ASR and SLU technologies that are employed in conversational interfaces, it is necessary to assume that the information captured by these components (and possibly also by sensors) may be uncertain or ambiguous. Error propagation from the ASR and SLU modules is one of the main problems that the dialog manager must try to detect and correct.

To deal with the uncertainty problem, the ASR and SLU components typically employ confidence scores attached to the frame slots, for example, real numbers in the range between 0 and 1. A confidence score that falls below a threshold indicates that the data item in the slot must be either confirmed or rejected by the system.

Two types of confirmation strategy are often employed: explicit confirmation and implicit confirmation. With explicit confirmation, the system generates an additional dialog turn to confirm the data item obtained from the previous user turn, as in the following example:

User: I want to know timetables from Madrid.
System: Do you want to leave from Madrid?
User: Yes.

The disadvantage of explicit confirmations is that the dialog tends to be lengthy due to these additional confirmation turns, and this makes the interaction less efficient and even excessively repetitive if all the data items provided by the user have to be confirmed.

The following is an example of an implicit confirmation:

User: I want to know timetables from Madrid.
System: What time do you want to leave from Madrid?

When the implicit confirmation strategy is used, the system includes some of the user's previous input in its next question. If the user answers the question directly, for example, in this case by stating a departure time, then it is assumed that the previous information about the destination is implicitly confirmed and no additional turns are required. However, it is the user's responsibility to make a correction if the system has misrecognized the information and this can lead to the user producing utterances that are beyond the scope of the ASR and SLU components, for example:

User: I want to know timetables from Madrid.
System: What time do you want to leave from Madrid?

User: No, I just wanted to know about times from Madrid but I might be departing from somewhere else depending on whether I have the use of the car next Friday.

These confirmation strategies are useful for avoiding misunderstandings, for example, when the system has understood something from its interaction with the user but is uncertain about how accurate it is. One related, but different situation is non-understanding, which occurs when the system has not been able to collect any data from its interaction with the user. In this case, two typical strategies for handling the error are to ask the user to repeat the input, or to ask for it to be rephrased.

In the case of multimodal conversational interfaces, the input information can also be ambiguous. For example, input made with a pen on a touch-sensitive screen can have three different purposes: pointing (as a substitute for the mouse), handwriting, and drawing. In order to address this problem, the system must employ some method to try to automatically decide the mode in which the pen is being used and/or employ an additional dialog turn to get a confirmation from the user about the intended mode.

A number of different approaches to DM have been developed within the research community and in industry (Lee et al. 2010; Wilks et al. 2011). These approaches can be classified into two main categories: handcrafted approaches using rules and statistical or data-driven approaches using machine learning methodologies. Hybrid approaches are also possible in which these two main approaches are combined. The following sections discuss approaches to DM.

10.3 Handcrafted Approaches to Dialog Management

One of the simplest DM strategies is finite state-based DM, in which a generic program implements the application with an interaction model based on finite state machines. This approach is usually confined to highly structured tasks in which system-directed initiative is used and the user's input is restricted to utterances within the scope of the ASR and SLU components (Barnard et al. 1999; Lee et al. 2010). This knowledge-based approach generally uses finite state automata with handcrafted rules. The user's actions determine the transitions between the system responses that constitute the nodes of the finite state automaton, and the user's responses to the system prompts are coded in recognition grammars.

Although this approach has been deployed in many practical applications because of its simplicity, these early applications only support a strict system-directed dialog interaction, in which at each turn the system directs the user by proposing a small number of choices for which there is a limited grammar or vocabulary to interpret the input. Directed dialog has been efficient in terms of

accuracy and cost of development. However, although libraries and dialog modules have been created that can be reused and adapted to different applications, the weakest point of this approach is its lack of versatility and poor domain portability (Acomb et al. 2007; Pieraccini et al. 2009).

Unlike the finite state approach, frame-based dialog managers do not have a predefined dialog path but use a frame structure comprised of one slot for piece of information that the system has to gather from the user (McTear 2004). The advantage of this approach is that the system can capture several data at once and the information can be provided in any order (more than one slot can be filled per dialog turn and in any order). The form interpretation algorithm (FIA), the basis for the VoiceXML standard, is an example of a model of frame-based dialog management (see Chap. 11). Using frames, it is possible to specify the whole topic of a dialog. A study by Lemon et al. (2001) is an example of a frame-based system, as is the COMIC DM system (Catizone et al. 2003). The core idea is that humans communicate to achieve goals and during the interaction the mental state of the speakers may change. Thus, frame-based dialog managers model dialog as a cooperation between the user and the system to reach common goals. Utterances are not considered as text strings but as dialog acts in which the user communicates their intentions.

A more advanced approach is Information State Theory, also known as Information State Update (ISU), introduced in Chap. 4 (Traum and Larsson 2003). The information state of a dialog represents the information needed to uniquely distinguish it from all others. It comprises the accumulated user interventions and previous dialog actions on which the next system response can be based. The information state is also sometimes known as the *conversation store*, *discourse context*, or *mental state*. In the information state approach, the main tasks for the dialog manager are to update the information state based on the observed user actions and based on this update to select the next system action as specified in the update rules.

Plan-based approaches take the view that humans communicate to achieve goals, including changes to the mental state of the listener. Plan-based theories of communicative action and dialog (e.g., Allen and Perrault 1980; Appelt 1985; Cohen and Levesque 1990) claim that the speaker's speech act is part of a plan and that it is the listener's task to identify and respond appropriately to this plan (Wilks et al. 2011). Plan-based approaches attempt to model this claim and explicitly represent the (global) goals of the task.

Conversational games theory (Carletta et al. 1995; Kowtko et al. 1993) uses techniques from both discourse grammars and plan-based approaches by including a goal or plan-oriented level in its structural approach. It can be used to model conversations between a human and a computer in a task-oriented dialog (Williams 1996). This approach deals with discourse phenomena such as side sequences and clarifications by allowing games to have another game embedded within them (Wilks et al. 2011).

Additionally, when it is necessary to execute and monitor operations in a dynamically changing application domain, an agent-based approach can be employed. A modular agent-based approach to DM makes it possible to combine

the benefits of different dialog control models, such as finite state-based dialog control and frame-based DM (Chu et al. 2005).

As previously discussed, in most settings, application developers, together with voice user interface (VUI) designers, typically handcraft DM strategies using rules and heuristics. As it is extremely challenging to anticipate every possible user input, handcrafting dialog management strategies is an error-prone process that needs to be iteratively refined and tuned, which requires considerable time and effort. The VoiceXML standard, which was introduced briefly in Chap. 4, is an example of the handcrafted approach that is used widely in industry to develop voice user interfaces. Chap. 11 provides an overview of VoiceXML along with exercises in how to build a simple dialog system using VoiceXML.

One of the main problems with handcrafted approaches to DM is that it is extremely challenging to anticipate every possible user input and design appropriate strategies to handle it. The process is error-prone and requires considerable time and effort to iteratively refine and tune the dialog strategies.

10.4 Statistical Approaches to Dialog Management

Machine learning approaches to DM try to reduce the effort and time required to handcraft DM strategies and, at the same time, facilitate the development of new dialog managers and their adaptation to deal with new domains. The application of machine learning approaches to DM strategy design is a rapidly growing research area. The main idea is to learn optimal strategies from corpora of real human–computer dialog data using automated "trial-and-error" methods instead of relying on empirical design principles (Young 2002).

Statistical approaches to DM present additional important advantages. Rather than maintaining a single hypothesis for the dialog state, they maintain a distribution over many hypotheses for the correct dialog state. In addition, statistical methodologies choose actions using an optimization process, in which a developer specifies high-level goals and the optimization works out the detailed dialog plan. Finally, statistical DM systems have shown, in research settings, more robustness to speech recognition errors, yielding shorter dialogs with higher task completion rates (Williams and Young 2007).

The main trend in this area is an increased use of data to improve the performance of the system. As described in Paek and Pieraccini (2008), there are three main aspects of spoken dialog interaction where the use of massive amounts of data can potentially improve the automation rate and ultimately the penetration and acceptance of speech interfaces in the wider consumer market. They are as follows:

- Task-independent behaviors (e.g., error correction and confirmation behavior).
- Task-specific behaviors (e.g., logic associated with certain customer care practices).
- Task interface behaviors (e.g., prompt selection).

Statistical models can be trained using corpora of human–computer dialogs with the goal of explicitly modeling the variability in user behavior that can be difficult to address by means of handwritten rules (Schatzmann et al. 2006). Additionally, it is possible to extend the strategy learned from the training corpus with handcrafted rules that include expert knowledge or specifications about the task (Suendermann and Pieraccini 2012; Laroche et al. 2008; Torres et al. 2008; Young et al. 2013).

The goal is to build systems that exhibit more robust performance, improved portability, better scalability, and easier adaptation to other tasks. However, model construction and parameterization are dependent on expert knowledge, and the success of statistical approaches is dependent on the quality and coverage of the models and data used for training (Schatzmann et al. 2006). Moreover, the training data must be correctly labeled for the learning process. The size of currently available annotated dialog corpora is usually too small to sufficiently explore the vast space of possible dialog states and strategies. Collecting a corpus with real users and annotating it requires considerable time and effort.

To address these problems, researchers have proposed alternative techniques that facilitate the acquisition and labeling of corpora, such as Wizard of Oz (Fraser and Gilbert 1991; Lane et al. 2004), bootstrapping (Fabbrizio et al. 2008; Abdennadher et al. 2007), active learning (Cohn et al. 1994; Venkataraman et al. 2005), automatic dialog act classification and labeling (O'Shea et al. 2012; Venkataraman et al. 2002), and user simulation (Schatzmann et al. 2006; Callejas et al. 2012).

Another relevant problem is how to deal with unseen situations, that is, situations that may occur during the dialog and that were not considered during training. To address this point, it is necessary to employ generalizable models in order to obtain appropriate system responses that enable the system to continue with the dialog in a satisfactory way.

Another difficulty is in the design of a good dialog strategy, which in many cases is far from being trivial. In fact, there is no clear definition of what constitutes a good dialog strategy (Schatzmann et al. 2006; Lemon and Pietquin 2012). Users are diverse, which makes it difficult to foresee which form of system behavior will lead to a quick and successful dialog completion, and speech recognition errors may introduce uncertainty about the user's intentions.

Statistical approaches to DM can be classified into three main categories: dialog modeling based on reinforcement learning (RL), corpus-based statistical dialog management, and example-based dialog management. Example-based approaches can be considered a specific case of corpus-based statistical dialog management, given that they usually perform dialog modeling by means of prepared dialog examples (Murao et al. 2003; Lee et al. 2009). These approaches assume that the next system action can be predicted when the dialog manager finds dialog examples that have a similar dialog state to the current dialog state (Lee et al. 2010). The best example is then selected from the candidate examples by calculating heuristic similarity measures between the current input and the example.

Hybrid approaches to DM combine statistical and rule-based approaches to try to reduce the amount of dialog data required for parameter estimation and to allow system designers to directly incorporate their expert domain knowledge into the

dialog models (Lison 2015). In the following sections, we provide a detailed description of reinforcement learning and corpus-based approaches.

10.4.1 Reinforcement Learning

The most recent research advances in reinforcement learning (RL) for building spoken conversational interfaces have been reviewed and summarized in a survey paper by Frampton and Lemon (2009). An earlier survey can be found in Schatzmann et al. (2006). See also Rieser and Lemon (2011).

The most widespread methodology for machine learning of dialog strategies involves modeling human–computer interaction as an optimization problem using Markov decision processes (MDPs) and reinforcement learning methods (Levin and Pieraccini 1997; Levin et al. 2000; Singh et al. 1999). The main drawback of this approach is that the large state space required for representing all the possible dialog paths in practical spoken conversational interfaces makes its direct representation intractable. In addition, while exact solution algorithms do exist, they do not scale to problems with more than a few states/actions (Young et al. 2010, 2013).

Partially observable MDPs (POMDPs) outperform MDP-based dialog strategies since they provide an explicit representation of uncertainty (Roy et al. 2000). This enables the dialog manager to avoid and recover from recognition errors by sharing and shifting probability mass between multiple hypotheses of the current dialog state.

Another disadvantage of the POMDP methodology is that the optimization process is free to choose any action at any time. As a result, there is no obvious way to incorporate domain knowledge or constraints such as business rules. In addition, in the worst case, spurious actions might be taken with real users, an especially serious concern if POMDP-based systems are going to handle financial or medical transactions. POMDP-based systems have been limited to small-scale problems, since the state space would be huge and exact POMDP optimization is again intractable (Young et al. 2010).

Formally, a partially observable MDP is defined as a tuple $\{S, A, T, R, O, Z, \lambda, b_0\}$ where

- `S` is a set of the system states;
- `A` is a set of actions that the system may take;
- `T` defines a transition probability $P(s'|s, a)$;
- `R` defines the immediate reward obtained from taking a particular action in a particular state $r(s, a)$;
- `O` is a set of possible observations that the system can receive from the world;
- `Z` defines the probability of a particular observation given the state and machine action $P(o'|s' a)$;
- λ is a geometric discount factor $0 \leq \lambda \leq 1$; and
- b_0 is an initial belief state $b_0(s)$.

The operation of a POMDP is as follows. At each moment, the system is in an unobserved state s. The system selects an action a_m, receives a reward r, and transits to a state (unobserved) s′, where s′ only depends on s and a_m. The system receives an observation o′, which depends on s′ and a_m. Although the observation allows the system to have some evidences about the state s in which the system is now, s is not exactly known, and b(s) (belief state) is defined to indicate the probability of the system being in the state s.

Based on b, the machine selects an action $a \in A$, receives a reward r(s, a), and transitions to state s′, which depends only on s and a. The machine then receives an observation $o' \in O$, which is dependent on s′ and a. In each moment, the probability of the system being in a specific state is updated taking into account o′ and a, as shown in Eq. 10.1.

$$\begin{aligned} b'(s') = P(s'|o', a, b) &= \frac{P(o'|s'_m, a_m, b)P(s'_m|a_m, b)}{P(o'|a_m, b)} \\ &= \frac{P(o'|s'_m, a_m, b)\sum_{s_m \in S_m} P(s'_m|a_m, b, s_m)P(s_m|a_m, b)}{P(o'|a_m, b)} \\ &= k \cdot P(o'|s', a)\sum\nolimits_{s \in S} P(s'|a, s)b(s) \end{aligned} \tag{10.1}$$

where k = P (o'|a, b) is a normalization constant (Kaelbling et al. 1998). At each time t, the system receives a reward $r(b_t, a_{m,t})$, which depends on b_t and the selected action $a_{m,t}$. The reward accumulated during the dialog is called a *return* and can be calculated by means of Eq. 10.2.

$$R = \sum_{t=0}^{\infty} \lambda^t R(b_t, a_{m,t}) = \sum_{t=0}^{\infty} \lambda^t \sum_{s \in S} b_t(s) r(s, a_{m,t}) \tag{10.2}$$

Each action $a_{m,t}$ is determined by the policy $\pi(b_t)$, and the construction of the POMDP model implies to find the strategy π^* which maximizes the return at every point b. Due to the vast space of possible belief states, however, the use of POMDPs for any practical system is far from straightforward. The optimal policy can be represented by a set of policy vectors where each vector v_i is associated with an action $a(i) \in A_m$ and $v_i(s)$ equals the expected value of taking action *a (i)* in state *s*. Given a complete set of policy vectors, the optimal value function and corresponding policy are computed as shown in Eq. 10.3.

$$V^{\pi^*}(b) = \max_i \{v_i, b\}$$

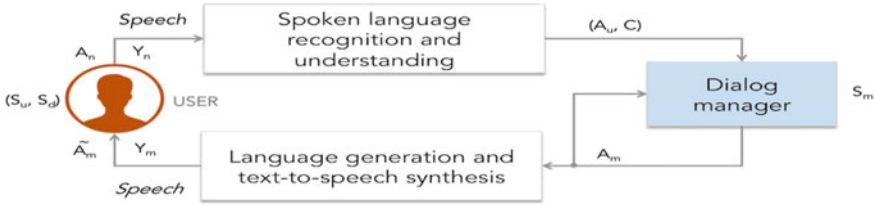

Fig. 10.1 Modeling a dialog system by means of POMDPs (Young et al. 2013)

and

$$V^{\pi^*}(b) = \max_i \{v_i, b\} \tag{10.3}$$

The application of a POMDP to model a conversational interface is based on the classical architecture of these systems as shown in Fig. 10.1. As this figure shows, the user has an internal state S_u corresponding to a goal to be accomplished and the dialog state S_d represents the previous history of the dialog. Based on the user's goal prior to each turn, the user decides some communicative action (also called an intention) A_u, expressed in terms of dialog acts and corresponding to an audio signal Y_u.

Then, the speech recognition and language understanding modules take the audio signal Y_u and generate the pair $\left(\widetilde{A}_u, C\right)$. This pair consists of an estimate of the user's action A_u and a confidence score that provides an indication of the reliability of the recognition and semantic interpretation results. This pair is then passed to the dialog model, which is in an internal state S_m and decides what action A_m the dialog system should take. This action is also passed back to the dialog manager so that S_m may track both user and machine actions. The language generator and the text-to-speech synthesizer take A_m and generate an audio response Y_m. The user listens to Y_m and attempts to recover A_m. As a result of this process, users update their goal state S_u and their interpretation of the dialog history S_d. These steps are then repeated until the end of the dialog.

One of the main challenges for conversational interfaces is that $\widetilde{A}_u$ usually contains recognition errors (i.e., $\widetilde{A}_u \neq A_u$). As a result, the user's action A_u, the user's state S_u, and the dialog history S_d are not directly observable and can never be known to the system with certainty. However, $\widetilde{A}_u$ and the confidence scores C provide evidence from which A_u, S_u, and S_d can be inferred.

Therefore, when using POMDPs to model a conversational interface, the POMDP state S_m expresses the unobserved state of the world and can naturally be factored into three distinct components: the user's goal S_u, the user's action A_u, and the dialog history S_d. Hence, the factored POMDP state S is defined as $S_m = (s_u,$

a_u, s_d). The belief state b is then a distribution over these three components: $s_m = b_s = b(s_u, a_u, s_d)$. The observation o is the estimate of the user dialog act $\widetilde{A}_u$. In the general case, this will be a set of N-best hypothesized user acts, each with an associated probability:

$$o = \left[\left(\tilde{a}_u^1, p_1\right), \left(\tilde{a}_u^2, p_2\right), \ldots, \left(\tilde{a}_u^N, p_N\right)\right] \tag{10.4}$$

where $p_n = P\left(\tilde{a}_u^N | o\right) \quad \text{for} \quad n = 1 \ldots N$.

The transition function for an SDS-POMDP follows directly by substituting the factored state into the regular POMDP transition function and making independence assumptions:

$$\begin{aligned} \boldsymbol{P}\left(\boldsymbol{s}'_{\boldsymbol{m}} | \boldsymbol{s}_{\boldsymbol{m}}, a_{\boldsymbol{m}}\right) &= \boldsymbol{P}\left(\boldsymbol{s}'_{\boldsymbol{u}}, \boldsymbol{a}'_{\boldsymbol{u}}, \boldsymbol{s}'_{\boldsymbol{d}} | \boldsymbol{s}_{\boldsymbol{u}}, \boldsymbol{a}_{\boldsymbol{u}}, \boldsymbol{s}_{\boldsymbol{d}}, a_{\boldsymbol{m}}\right) \\ &= \boldsymbol{P}\left(\boldsymbol{s}'_{\boldsymbol{u}} | \boldsymbol{s}_{\boldsymbol{u}}, \boldsymbol{a}_{\boldsymbol{m}}\right) \boldsymbol{P}\left(\boldsymbol{a}'_{\boldsymbol{u}} | \boldsymbol{s}'_{\boldsymbol{u}}, \boldsymbol{a}_{\boldsymbol{m}}\right) \boldsymbol{P}\left(\boldsymbol{s}'_{\boldsymbol{d}} | \boldsymbol{s}'_{\boldsymbol{u}}, \boldsymbol{a}'_{\boldsymbol{u}}, \boldsymbol{s}_{\boldsymbol{d}}, \boldsymbol{a}_{\boldsymbol{m}}\right) \end{aligned} \tag{10.5}$$

The observation model is obtained by making similar reasonable independence assumptions regarding the observation function, giving

$$P\left(o' | s'_m, a_m\right) = P\left(o' | s'_u, a'_u, s'_d, a_m\right) = P\left(o' | a'_u\right) \tag{10.6}$$

The above factoring simplifies the belief update equation as shown in Eq. 10.7.

$$\begin{aligned} b'\left(s'_u, a'_u, s'_d\right) = k \times & \underbrace{P\left(o' | a'_u\right)}_{\text{Observation model}} \underbrace{P\left(a'_u | s'_u, a_m\right)}_{\text{User action model}} \sum_{s_u} \underbrace{P\left(s'_u | s_u, a_m\right)}_{\text{User goal model}} \\ & \times \sum_{s_d} \underbrace{P\left(s'_d | s'_u, a'_u, s_d, a_m\right)}_{\text{Dialog model}} b\left(s_u, s_d\right) \end{aligned} \tag{10.7}$$

As shown in the previous equation, the probability distribution for a'_u is called the user action model. It allows the observation probability to be scaled by the probability that the user would speak a'_u given the goal s'_u and the last system prompt a_m. The user goal model determines the probability of the user goal switching from s_u to s'_u following the system prompt a_m. Finally, the dialog model enables information relating to the dialog history to be maintained such as grounding and focus.

10.4.1.1 Reinforcement Learning: Some Problems and Some Solutions

Scaling the dialog model to handle real-world problems remains a significant challenge for RL-based systems, given that the complexity of a POMDP grows with the number of user goals, and optimization quickly becomes intractable. The summary POMDP method (Young et al. 2010) provides a way to scale up the POMDP model for so-called slot-filling spoken dialog systems. In this approach,

the belief state and actions are mapped down to a summarized form where optimization becomes tractable.

The original belief space and actions are called *master space* and *master actions*, while the summarized versions are called *summary space* and *summary actions*. The updated belief state b is then mapped into a summary state $\tilde{b}$, where an optimized dialog policy is applied to compute a new summary machine action $\tilde{a}_m$. The summary machine action is then mapped back into the master space where it is converted to a specific machine dialog act a_m.

The optimization of the policy in these two spaces is usually carried out using techniques such as *point-based value iteration* or *Q-learning*, in combination with a user simulator. Q-learning is a technique for online learning where a sequence of sample dialogs is used to estimate the Q functions for each state and action. The optimal action for each point p is given by

$$\bar{a}_p = \underset{\bar{a}}{\operatorname{argmax}}\, \overline{Q}(a, p) \tag{10.8}$$

Given that a good estimate of the true Q-value can be obtained if sufficient dialogs are completed, user simulation is usually introduced to reduce the time-consuming and expensive task of obtaining these dialogs with real users. Simulation is usually done at a semantic dialog act level to avoid having to reproduce the variety of user utterances at the word or acoustic levels.

Agenda-based state representations, like the one described in (Thomson et al. 2007), factor the user state into an agenda A and a goal G. The goal G consists of constraints C that specify the detailed goal of the dialog and requests R that specify the desired pieces of information.

The user agenda A is a stack-like structure containing the pending user dialog acts that are needed to elicit the information specified in the goal. At the beginning of each dialog, a new goal G is randomly selected. Then, the goal constraints C are converted into user and system inform acts (a_u and a_m acts) and the requests R into request acts. A bye act is added at the bottom of the agenda to close the dialog once the goal has been fulfilled. The agenda is ordered according to priority, with A[N] denoting the top item and A[1] denoting the bottom item. As the dialog progresses, the agenda and goal are dynamically updated and acts are selected from the top of the agenda to form user acts a_u.

Young et al. (2010) present an approach that scales the POMDP framework for the implementation of practical spoken conversational interfaces by defining two state spaces. Approximate algorithms have also been developed to overcome the intractability of exact algorithms, but even the most efficient of these techniques such as point-based value iteration (PBVI) cannot scale to the many thousands of states required by a statistical dialog manager (Williams et al. 2006).

Composite summary point-based value iteration (CSPBVI) has suggested the use of a small summary space for each slot where PBVI policy optimization can be applied. However, policy learning in this technique can only be performed off-line, i.e., at design time, because policy training requires an existing accurate model of user behavior. An alternative technique for online training based on Q-learning is presented in Thomson et al. (2007), which allows the system to adapt to real users as new dialogs are recorded. This technique does not require any model of user behavior, so user simulation techniques are proposed to iteratively learn the dialog model.

Other authors have combined conventional dialog managers with a fully observable Markov decision process (Singh et al. 2002; Heeman 2007), or proposed using multiple POMDPs and selecting actions using handcrafted rules (Williams et al. 2006). In Williams (2008), the robustness of the POMDP approach is combined with the developer control available in conventional approaches: The (conventional) dialog manager and POMDP run in parallel, but the dialog manager is augmented so that it outputs one or more allowed actions at each time step. The POMDP then chooses the best action from this limited set. Results from a real voice dialer application show that adding the POMDP machinery to a standard dialog system yields a significant improvement.

Crook et al. (2014) describe an evaluation of a POMDP-based spoken dialog system using crowd-sourced calls with real users. The evaluation compares a "hidden information state" POMDP system that uses a handcrafted compression of the belief space with the same system using instead an automatically computed belief space compression.

In Tetreault and Litman (2008), the authors aimed to evaluate the best state-space representations so that RL can be used to find an optimal dialog policy. The authors presented three metrics for the tutoring domain and ways to build confidence intervals for model switching. In the work reported in Gašić et al. (2011), online optimization of dialog policy was conducted in spoken dialog systems via live interaction with human subjects.

Jurčíček et al. (2012) presented two RL algorithms for learning the parameters of a dialog model. The Natural Belief Critic algorithm is designed to optimize the model parameters while the policy is kept fixed. The Natural Actor and Belief Critic algorithm jointly optimizes both the model and the policy parameters. The algorithms were evaluated on a statistical dialog system for the tourist information domain modeled as a POMDP. The experiments indicated that model parameters estimated to maximize the expected reward function provide improved performance compared to the baseline handcrafted parameters.

Thomson and Young (2010) used expectation–propagation (EP) to infer the unobserved dialog state together with the model parameters. The main advantage of this algorithm is that it is an off-line method and it does not rely on annotated data. However, it requires the model to be generative (i.e., the observations must be conditioned on the dialog state).

Wierstra et al. (2010) used recurrent neural networks (RNN) to approximate the policy. This method selects a new system action based on the accumulated

information in the internal memory and the last observation. Png and Pineau (2011) presented a framework based on a Bayes-adaptive POMDP algorithm to learn an observation model. In this work, a dialog model was factored into a transition model between hidden dialog states and an observation model, and only learning of the observation model was considered.

Lopes et al. (2015) have very recently presented a data-driven approach to improve the performance of SDSs by automatically finding the most appropriate terms to be used in system prompts. Speakers use one another's terms (entrain) when trying to create common ground during a spoken dialog. Those terms are commonly called *primes*, since they influence the interlocutors' linguistic decision-making. The proposed approach emulates human interaction, with a system built to propose primes to the user and accept the primes that the user proposes. Live tests with this method show that the use of on-the-fly entrainment reduces out-of-vocabulary and word error rate and also increases the number of correctly transferred concepts.

Lison (2015) has also recently presented a modeling framework for DM based on the concept of probabilistic rules, which are defined as structured mappings between logical conditions and probabilistic effects. Probabilistic rules are able to encode the probability and utility models employed in DM in a compact and human-readable form. As a consequence, they can reduce the amount of dialog data required for parameter estimation and allow system designers to directly incorporate their expert domain knowledge into the dialog models.

Other interesting approaches for statistical DM are based on modeling the system by means of Hidden Markov models (HMMs) (Cuayáhuitl et al. 2005) or using Bayesian networks (Paek and Horvitz 2000; Meng et al. 2003).

10.4.2 Corpus-Based Approaches

Griol et al. (2014) describe a corpus-based approach to DM based on the estimation of a statistical model from the sequences of the system and user dialog acts obtained from a set of training data. The next system response is selected by means of a classification process that considers the complete history of the dialog.

Another main characteristic is the inclusion of a data structure that stores the information provided by the user. The main objective of this structure is to easily encode the complete information related to the task provided by the user during the dialog history and then to consider the specific semantics of the task and include this information in the proposed classification process.

In order to control the interactions with the user, the proposed dialog manager represents dialogs as a sequence of pairs (A_i, U_i), where A_i is the output of the dialog system (the system answer) at time i and U_i is the semantic representation of the user turn (the result of the understanding process of the user input) at time i; both expressed in terms of dialog acts (Griol et al. 2008). Each dialog is represented by:

$$(A_1, U_1), \ldots, (A_i, U_i), \ldots, (A_n, U_n)$$

where A_1 is the greeting turn of the system and U_n is the last user turn. We refer to a pair (A_i, U_i) as S_i, the dialog sequence at time i.

In this framework, we consider that, at time i, the objective of the dialog manager is to find the best system answer A_i. This selection is a local process for each time i that takes into account the previous history of the dialog, that is to say, the sequence of states of the dialog preceding time i:

$$\widehat{A}_i = \underset{A_i \in \mathcal{A}}{\operatorname{argmax}} P(A_i \mid S_1, \ldots, S_{i-1}) \tag{10.9}$$

where set A contains all the possible system answers.

Following Eq. 10.9, the dialog manager selects the next system prompt by taking into account the sequence of previous pairs (A_i, U_i). The main problem with resolving this equation is that the number of possible sequences of states is usually very large. To solve the problem, we define a data structure in order to establish a partition in this space, i.e., in the history of the dialog preceding time i. This data structure, which we call *Dialog Register* (DR), contains the information provided by the user throughout the previous history of the dialog.

After applying the above considerations and establishing the equivalence relation in the histories of the dialogs, the selection of the best A_i is given by:

$$\widehat{A}_i = \underset{A_i \in \mathcal{A}}{\operatorname{argmax}} P(A_i \mid DR_{i-1}, S_{i-1}) \tag{10.10}$$

Each user turn supplies the system with information about the task; i.e., the user asks for a specific concept and/or provides specific values for certain attributes. However, a user turn can also provide other kinds of information, such as task-independent information (for instance, `Affirmation`, `Negation`, and `Not-Understood` dialog acts). This kind of information implies some decisions that are different from simply updating the DR_{i-1}. Hence, for the selection of the best system response A_i, we take into account the DR that results from turn 1 to turn $i - 1$, and we explicitly consider the last state S_{i-1}.

We propose solving Eq. 10.10 using a classification process, in which every dialog situation (i.e., each possible sequence of dialog acts) is classified taking into account a set of classes C, in which a class contains all the sequences that provide the same set of system actions (responses). The objective of the dialog manager at each moment is to select a class of this set $c \in C$, so that the system answer is the one associated with the selected class.

The classification function can be defined in several ways. Griol et al. (2014) propose the use of a multilayer perceptron (MLP) (Rumelhart et al. 1986), where the input layer holds the input pair (DR_{i-1}, S_{i-1}) corresponding to the Dialog Register and the state. The values of the output layer can be seen as an

approximation of the a posteriori probability of the input belonging to the associated class $c \in C$.

As stated before, the DR contains information about concepts and values for the attributes provided by the user throughout the previous history of the dialog. For the dialog manager to determine the next answer, the exact values of the attributes are assumed to be not significant. They are important for accessing databases and for constructing the output sentences of the system. However, the only information necessary to predict the next action by the system is the presence or absence of concepts and attributes. Therefore, the codification proposed for each slot in the DR is in terms of three values, {0, 1, 2}, according to the following criteria:

- (0) The concept is unknown, or the value of the attribute is not given.
- (1) The concept or attribute is known with a confidence score that is higher than a given threshold.
- (2) The concept or attribute has a confidence score that is lower than the given threshold.

To decide whether the state of a certain value in the DR is 1 or 2, the system employs confidence measures provided by the ASR and SLU modules (Torres et al. 2005).

The previously described process allows every task to be modeled based only on the information provided by the user in the previous turns and its own model. In other dialog systems, the dialog manager generates the next system response taking also into account the information generated by the module that controls the application (that is denoted as the application manager (AM)). For example, the AM can validate restrictions, apply privacy policies, or carry out computations that define the next system response (for instance, selecting a different system action depending on the result of a query to the databases of the application). Thus, the output of this module has to be taken into account for the selection of the best system action.

For this reason, for this kind of task, two phases are proposed for the selection of the next system turn. In the first phase, the information contained in the DR and the last state S_{i-1} are considered to select the best request to be made to the AM:

$$\widehat{A}_i = \underset{A_{1_i} \in \mathcal{A}_1}{\operatorname{argmax}} P(A_i \mid DR_{i-1}, S_{i-1}) \tag{10.11}$$

where A_1 is the set of possible requests to the AM.

In the second phase, the system answer $\widetilde{A}_2$ is generated taking into account $\widetilde{A}_1$ and the information provided by the AM (AM_i):

$$\widehat{A}_{2_i} = \underset{A_{2_i} \in \mathcal{A}_2}{\operatorname{argmax}} P(A_i | AM_i, A_{1_i}) \tag{10.12}$$

where $\widetilde{A}_2$ is the set of possible system answers.

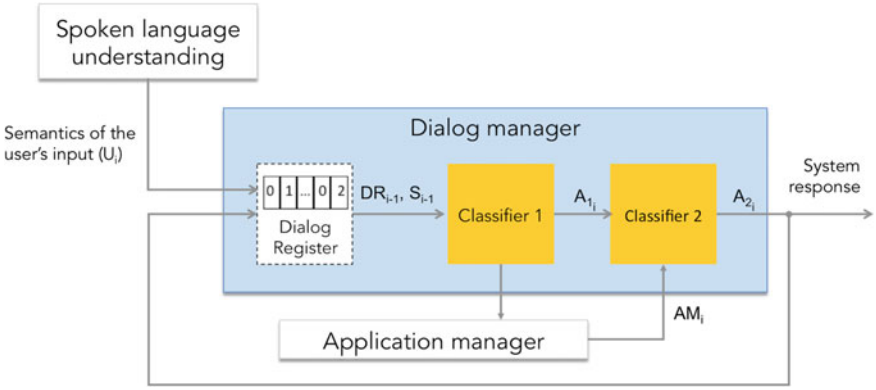

Fig. 10.2 Scheme of the architecture proposed in the corpus-based DM methodology

Figure 10.2 shows the scheme proposed for the development of the dialog manager for this kind of task, detailing the two phases described for the generation of the system response. The use of two MLPs is proposed to deal with the specific information defined for each phase.

The AM makes it possible to consider specific requisites (e.g., special requirements, policies, or specific routines) that endow conversational interfaces with a more sophisticated behavior that is different from only requiring information from the user and checking or updating a repository. This phase also makes it possible for systems to deal with specific cases for the different attributes, given that the exact values for each attribute are considered to access the data repositories. In addition, the statistical dialog model supports user adaptation, which makes it suitable for different application domains with varying degrees of complexity.

10.5 Summary

Given the current state of the dialog, the principal role of the dialog manager is to choose an action that will result in a change of dialog state. The strategy followed by the dialog manager, sometimes referred to as the policy, should be designed to enable successful, efficient, and natural conversations.

This is a challenging goal, and in most commercially deployed conversational interfaces, a human designer handcrafts the dialog manager. This handcrafted approach is limited for several reasons: it is not always easy to specify the optimal action at each state of the dialog; a dialog behavior that is generic and static is usually assumed for the entire user population; designing such strategies is labor-intensive, especially for large systems.

Machine learning approaches to DM try to reduce the effort and time required by handcrafted DM strategies; they isolate domain knowledge from the dialog strategy;

and they facilitate the development of new dialog managers and their adaptation to new domains.

DM is discussed further in Chap. 11 where we provide practical exercises related to the application of rule-based and statistical DM techniques for a specific task. Chap. 17 will discuss the most relevant approaches for the evaluation of DM.

Further Reading

Wilks et al. (2011) present a detailed survey of DM approaches and architectures, along with practical examples of dialog systems developed using these approaches and architectures. Lee et al. (2010) present a detailed survey covering design issues and approaches to DM and techniques for modeling. The paper also explains the use of user simulation techniques for the automatic evaluation of spoken conversational interfaces.

Some recent research advances in RL for building SDSs were reviewed and summarized in a survey paper by Frampton and Lemon (2009). An earlier survey can be found in Schatzmann et al. (2006). See also Rieser and Lemon (2011), Lemon and Pietquin (2012), and Thomson (2013). Young et al. (2013) provide an overview of the state of the art in the development of POMDP-based spoken dialog systems.

Meena et al. (2014) summarize the main approaches to turn taking in human conversations. They also explore a range of automatically extractable features for online use, covering prosody, lexicosyntax, and context, and different classes of learning algorithms for turn taking in human–machine conversations.

References

Abdennadher S, Aly M, Bühler D, Minker W, Pittermann J (2007) Becam tool—a semi-automatic tool for bootstrapping emotion corpus annotation and management. In: Proceedings of the international conference on spoken language processing (Interspeech'2007), Antwerp, Belgium, 27–31 Aug 2007, pp 946–949. http://met.guc.edu.eg/Repository/Faculty/Publications/69/Paper.pdf

Acomb K, Bloom J, Dayanidhi K, Hunter P, Krogh P, Levin E, Pieraccini R (2007) Technical support dialog systems: issues, problems, and solutions. In: Proceedings of the NAACL-HLT-Dialog'07 workshop on bridging the gap: Academic and Industrial Research in Dialog Technologies, Rochester, NY, USA, 26 Apr 2007, pp 25–31. http://dl.acm.org/citation.cfm?id=1556332&CFID=585421472&CFTOKEN=72903197

Allen JF, Perrault CR (1980) Analyzing intentions in dialogs. Artif Intell 15(3):143–178. doi:10.1016/0004-3702(80)90042-9

Appelt DE (1985) Planning English sentences. Cambridge University Press, Cambridge. doi:10.1017/CBO9780511624575

Barnard E, Halberstadt A, Kotelly C, Phillips M (1999) A consistent approach to designing spoken-dialog Systems. In: Proceedings of the IEEE workshop on automatic speech recognition and understanding (ASRU'99), Keystone, Colorado, USA, pp 1173–1176

Callejas Z, Griol D, Engelbrecht K, López-Cózar R (2012) A clustering approach to assess real user profiles in spoken dialogue systems. In: Mariani J, Rosset S, Garnier-Rizet M, Devilliers L (eds) Natural language interaction with robots: putting spoken dialog systems into practice. Springer, New York, pp 327–334. doi:10.1007/978-1-4614-8280-2_29

Carletta JC, Isard A, Isard S, Kowtko J, Doherty-Sneddon G, Anderson A (1995) The coding of dialog structure in a corpus. In: Andernach T, van de Burgt SP, van der Hoeven GF (eds) Proceedings of the Twente workshop on language technology: corpus-based approaches to dialogue modelling, University of Twente, Netherlands, June 1995

Catizone R, Setzer A, Wilks Y (2003) Multimodal dialogue management in the COMIC project. In: Jokinen K, Gamback B, Black W, Catizone R, Wilks Y (eds) Proceedings of the 2003 EACL workshop on dialogue systems: interaction, adaptation and styles of management, Budapest, Hungary, 13–14 Apr 2003. http://aclweb.org/anthology/W/W03/W03-2705.pdf

Chu S, O'Neill I, Hanna P, McTear M (2005) An approach to multistrategy dialog management. In: Proceedings of the 9th international conference on spoken language processing (Interspeech2005), Lisbon, Portugal, pp 865–868. http://www.isca-speech.org/archive/archive_papers/interspeech_2005/i05_0865.pdf

Cohen P, Levesque H (1990) Rational interaction as the basis for communication. In: Cohen P, Morgan J, Pollack M (eds) Intentions in communication. MIT Press, Cambridge, MA, pp 221–256. https://www.sri.com/work/publications/rational-interaction-basis-communication. Accessed 20 Jan 2016

Cohn DA, Atlas L, Ladner R (1994) Improving generalization with active learning. Mach Learn 15 (2):201–221. doi:10.1007/BF00993277

Crook PA, Keizer S, Wang Z, Tang W, Lemon O (2014) Real user evaluation of a POMDP spoken dialog system using automatic belief compression. Comput Speech Lang 28(4):873–887. doi:10.1016/j.csl.2013.12.002

Cuayáhuitl H, Renals S, Lemon O, Shimodaira H (2005) Human-computer dialogue simulation using Hidden Markov models. In: Proceedings of the IEEE workshop on automatic speech recognition and understanding (ASRU2005), San Juan, Puerto Rico, 27 Nov 2005, pp 290–295. doi:10.1109/ASRU.2005.1566485

Fabbrizio GD, Tur G, Hakkani-Tür D, Gilbert M, Renger B, Gibbon D, Liu Z, Shahraray B (2008) Bootstrapping spoken dialogue systems by exploiting reusable libraries. Nat Lang Eng 14 (3):313–335. doi:10.1017/S1351324907004561

Frampton M, Lemon O (2009) Recent research advances in reinforcement learning in spoken dialog systems. Knowl Eng Rev 24(4):375–408. doi:10.1017/S0269888909990166

Fraser M, Gilbert G (1991) Simulating speech systems. Comput Speech Lang 5(1):81–99. doi:10.1016/0885-2308(91)90019-M

Gašić M, Jurčíček F, Thomson B, Yu K, Young S (2011) On-line policy optimisation of spoken dialog systems via live interaction with human subjects. In: Proceedings of IEEE workshop on automatic speech recognition and understanding (ASRU), Waikoloa, Hawaii, 11–15 Dec 2011, pp 312–317. doi:10.1109/ASRU.2011.6163950

Griol D, Hurtado LF, Segarra E, Sanchis E (2008) A statistical approach to spoken dialog systems design and evaluation. Speech Commun 50(8–9):666–682. doi:10.1016/j.specom.2008.04.001

Griol D, Callejas Z, López-Cózar R, Riccardi G (2014) A domain-independent statistical methodology for dialog management in spoken dialog systems. Comput Speech Lang 28 (3):743–768. doi:10.1016/j.csl.2013.09.002

Heeman, P (2007) Combining reinforcement learning with information-state update rules. In: Proceedings of the 8th annual conference of the North American chapter of the Association for Computational Linguistics (HLT-NAACL2007), Rochester, New York, USA, 22–27 Apr 2007. http://aclweb.org/anthology/N07-1034

Jurčíček F, Thomson B, Young S (2012) Reinforcement learning for parameter estimation in statistical spoken dialog systems. Comput Speech Lang 26(3):168–192. doi:10.1016/j.csl.2011.09.004

Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artif Intell 101(1–2):99–134. doi:10.1016/s0004-3702(98)00023-x

Kowtko JC, Isard SD, Doherty, GM (1993) Conversational games within dialogue. Human Communication Research Centre, University of Edinburgh, (HCRC/RP-31). doi:10.1.1.52.5350

Lane I, Ueno S, Kawahara T (2004) Cooperative dialogue planning with user and situation models via example-based training. In: Proceedings of workshop on man-machine symbiotic systems, Kyoto, Japan, 23–24 Nov 2004, pp 93–102

Laroche R, Putois G, Bretier P, Young S, Lemon O (2008) Requirements analysis and theory for statistical learning approaches in automaton-based dialogue management. CLASSiC Project Deliverable 1.1.1. Edinburgh University, Edinburgh, UK. http://www.classic-project.org/deliverables/d1.1.1.pdf

Lee C, Jung S, Kim S, Lee G (2009) Example-based dialog modeling for practical multi-domain dialog system. Speech Commun 51(5):466–484. doi:10.1016/j.specom.2009.01.008

Lee CJ, Jung SK, Kim KD, Lee DH, Lee GG (2010) Recent approaches to dialog management for spoken dialog systems. J Comput Sci Eng 4(1):1–22. doi:10.5626/JCSE.2010.4.1.001

Lemon O, Pietquin O (eds) (2012) Data-driven methods for adaptive spoken dialog systems: computational learning for conversational interfaces. Springer, New York. doi:10.1007/978-1-4614-4803-7

Lemon O, Bracy A, Gruenstein A, Peters S (2001) The Witas multimodal dialog system I. In: Proceedings of the 7th Eurospeech conference on speech communication and technology (INTERSPEECH'01), Aalborg, Denmark, 3–7 Sept 2001, pp 1559–1562. http://www.isca-speech.org/archive/eurospeech_2001/e01_1559.html

Levin E, Pieraccini R (1997) A stochastic model of human-machine interaction for learning dialog strategies. In: Proceedings of the 5th European conference on speech communications and technology (Eurospeech1997), Rhodes, Greece, pp 1883–1886. http://www.isca-speech.org/archive/eurospeech_1997/e97_1883.html

Levin E, Pieraccini R, Eckert W (2000) A stochastic model of human-machine interaction for learning dialog strategies. IEEE T on Speech Audi P 8(1):11–23. doi:10.1109/89.817450

Lison P (2015) A hybrid approach to dialogue management based on probabilistic rules. Comput Speech Lang 34(1):232–255. doi:10.1016/j.csl.2015.01.001

Lopes J, Eskenazi M, Trancoso I (2015) From rule-based to data-driven lexical entrainment models in spoken dialog systems. Comput Speech Lang 31(1):87–112. doi:10.1016/j.csl.2014.11.007

McTear M (2004) Spoken dialogue technology: toward the conversational user interface. Springer, New York. doi:10.1007/978-0-85729-414-2

Meena R, Skantze G, Gustafson J (2014) Data-driven models for timing feedback responses in a pap task dialogue system. Comput Speech Lang 28(4):903–922. doi:10.1016/j.csl.2014.02.002

Meng HH, Wai C, Pieraccini R (2003) The use of belief networks for mixed-initiative dialog modeling. IEEE Trans Speech Audio Process 11(6):757–773. doi:10.1109/TSA.2003.814380

Murao HK, Kawaguchi N, Matsubara S, Ymaguchi Y, Inagaki Y (2003) Example-based spoken dialogue system using WOZ system sog. In: Proceedings of the 4th SIGDIAL workshop on discourse and dialogue, Sapporo, Japan, 5–6 July 2003, pp 140–148. http://www.aclweb.org/anthology/W/W03/W03-2112.pdf

O'Shea J, Bandar Z, Crockett K (2012) A multi-classifier approach to dialog act classification using function words. In: Nguyen NT (ed) Transactions on computational collective intelligence VII. Lecture notes in computer science, vol 7270, pp 119–143. doi:10.1007/978-3-642-32066-8_6

Paek T, Horvitz E (2000) Conversation as action under uncertainty. In: Proceedings of the 16th conference on uncertainty in artificial intelligence, Stanford, CA, USA, pp 455–464. http://arxiv.org/pdf/1301.3883.pdf

Paek T, Pieraccini R (2008) Automating spoken dialog management design using machine learning: an industry perspective. Speech Commun 50:716–729. doi:10.1016/j.specom.2008.03.010

Pieraccini R, Suendermann D, Dayanidhi K, Liscombe J (2009) Are we there yet? Research in commercial spoken dialog systems. In: Matoušek V, Mautner P (eds) Text, speech and dialogue: 12th international conference, TSD 2009, Pilsen, Czech Republic, 13–17 Sept 2009, pp 3–13. doi:10.1007/978-3-642-04208-9_3

Png S, Pineau J (2011) Bayesian reinforcement learning for POMDP-based dialogue systems. In: Proceedings of international conference on acoustics, speech and signal processing (ICASSP2011), Prague, Czech Republic, 22–27 May 2011, pp 2156–2159. doi:10.1109/ICASSP.2011.5946754

Rieser V, Lemon O (2011) Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation. Springer, New York. doi:10.1007/978-3-642-24942-6

Roy N, Pineau J, Thrun S (2000) Spoken dialogue management using probabilistic reasoning. In: Proceedings of the 38th annual meeting of the Association for Computational Linguistics (ACL2000), Hong Kong, China, 1–8 Oct 2000. https://aclweb.org/anthology/P/P00/P00-1013.pdf

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumerhart DE, McClelland JL (eds) Parallel distributed processing: explorations in the microstructure of cognition, vol 1. MIT Press, Cambridge, pp 318–362. http://dl.acm.org/citation.cfm?id=104293

Schatzmann J, Weilhammer K, Stuttle M, Young S (2006) A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. Knowl Eng Rev 21 (2):97–126. doi:10.1017/s0269888906000944

Singh S, Kearns M, Litman D, Walker M (1999) Reinforcement learning for spoken dialog systems. In: Proceedings of neural information processing systems (NIPS 1999), Denver, USA, pp 956–962. http://papers.nips.cc/paper/1775-reinforcement-learning-for-spoken-dialogue-systems.pdf

Singh S, Litman D, Kearns M, Walker M (2002) Optimizing dialogue management with reinforcement learning: experiments with the NJFun system. J Artif Intell Res 16:105–133. doi:10.1613/jair.859

Suendermann D, Pieraccini R (2012) One year of contender: what have we learned about assessing and tuning industrial spoken dialog systems? In: Proceedings of the NAACL-HLT workshop on future directions and needs in the spoken dialog community: tools and data (SDCTD 2012), Montreal, Canada, 7 June 2012, pp 45–48. http://www.aclweb.org/anthology/W12-1818 Accessed 20 Jan 2016

Tetreault JR, Litman D (2008) A reinforcement learning approach to evaluating state representations in spoken dialogue systems. Speech Commun 50(8–9):683–696. doi:10.1016/j.specom.2008.05.002

Thomson B (2013) Statistical methods for spoken dialog management. Springer theses. Springer, New York. doi:10.1007/978-1-4471-4923-1

Thomson B, Young S (2010) Bayesian update of dialog state: a POMDP framework for spoken dialogue systems. Comput Speech Lang 24(4):562–588. doi:10.1016/j.csl.2009.07.003

Thomson B, Schatzmann J, Weilhammer K, Ye H, Young S (2007) Training a real-world POMDP-based dialogue system. In: Proceedings of NAACL-HLT-Dialog'07 workshop on bridging the gap: academic and industrial research in dialog technologies, Rochester, NY, USA, pp 9–16. http://dl.acm.org/citation.cfm?doid=1556328.1556330

Torres F, Hurtado LF, García F, Sanchis E, Segarra E (2005) Error handling in a stochastic dialog system through confidence measures. Speech Commun 45:211–229. doi:10.1016/j.specom.2004.10.014

Torres F, Sanchis E, Segarra E (2008) User simulation in a stochastic dialog system. Comput Speech Lang 22(3):230–255. doi:10.1016/j.csl.2007.09.002

Traum DR, Larsson S (2003) The information state approach to dialog management. In: Smith R, Kuppevelt J (eds) Current and new directions in discourse and dialog. Kluwer Academic Publishers, Dordrecht, pp 325–353. doi:10.1007/978-94-010-0019-2_15

Venkataraman A, Stolcke A, Shriberg E (2002) Automatic dialog act labeling with minimal supervision. In: Proceedings of the 9th australian international conference on speech science and technology, Melbourne, Australia, 2–5 Dec 2002. https://www.sri.com/sites/default/files/publications/automatic_dialog_act_labeling_with_minimal.pdf

Venkataraman A, Liu Y, Shriberg E, Stolcke A (2005) Does active learning help automatic dialog act tagging in meeting data. In: Proceedings of interspeech-2005, Lisbon, Portugal, 4–8 Sept 2005, pp 2777–2780. http://www.isca-speech.org/archive/interspeech_2005/i05_2777.html

Wierstra D, Förster A, Peters J, Schmidhuber J (2010) Recurrent policy gradients. Logic J IGPL 18 (5):620–634. doi:10.1093/jigpal/jzp049

Wilks Y, Catizone R, Worgan S, Turunen M (2011) Some background on dialogue management and conversational speech for dialogue systems. Comput Speech Lang 25(2):128–139. doi:10.1016/j.csl.2010.03.001

Williams S (1996) Dialogue management in mixed-initiative, cooperative, spoken language system. Proceedings of 11th twente workshop on language technology (TWLT11) dialogue management in natural language systems, Enschade, Netherlands. doi: http://users.mct.open.ac.uk/sw6629/Publications/twlt96.pdf

Williams, JD (2008) The best of both worlds: Unifying conventional dialog systems and POMDPs. In: Proceedings of the international conference on spoken language processing (InterSpeech-2008), Brisbane, Australia, 22–16 Sept 2016, pp 1173–1176. http://www.isca-speech.org/archive/interspeech_2008/i08_1173.html

Williams JD, Young S (2007) Partially observable Markov decision processes for spoken dialog systems. Comput Speech Lang 21(2):393–422. doi:10.1016/j.csl.2006.06.008

Williams JD, Poupart P, Young S (2006) Partially observable Markov decision processes with continuous observations for dialog management. In: Dybkær L, Minker W (eds) Recent trends in discourse and dialogue. Springer, New York, PP 191–217. doi: 10.1007/978-1-4020-6821-8_8

Young S (2002) Talking to machines (statistically speaking). In: Proceedings of the 7th international conference on spoken language processing, Denver, Colorado, USA, 16–20 sept 2002, pp 9–16. http://www.isca-speech.org/archive/archive_papers/icslp_2002/i02_0009.pdf

Young S, Gašić M, Keizer S, Mairesse F, Schatzmann J, Thomson B, Yu K (2010) The Hidden Information State model: a practical framework for POMDP-based spoken dialogue management. Comp Speech Lang 24(2):150–174. doi:10.1016/j.csl.2009.04.001

Young S, Gašić M, Thomson B, Williams J (2013) POMDP-based statistical spoken dialog systems: a review. In: Proceedings of the IEEE 101(5), Montreal, Canada, pp 1160–1179. doi:10.1109/JPROC.2012.2225812