CHAPTER

30 Advanced Dialog Systems

A famous burlesque routine from the turn of the last century plays on the difficulty of conversational understanding by inventing a baseball team whose members have confusing names:

C: *I want you to tell me the names of the fellows on the St. Louis team.*
A: *I'm telling you. Who's on first, What's on second, I Don't Know is on third.*
C: *You know the fellows' names?*
A: *Yes.*
C: *Well, then, who's playing first?*
A: *Yes.*
C: *I mean the fellow's name on first.*
A: *Who.*
C: *The guy on first base.*
A: *Who is on first.*
C: *Well what are you askin'* me *for?*
A: *I'm not asking you – I'm telling you. Who is on first.*

Who's on First – Bud Abbott and Lou Costello's version of an old burlesque standard.

Of course outrageous names of baseball players are not a normal source of difficulty in conversation. What this famous comic conversation is pointing out is that understanding and participating in dialog requires knowing whether the person you are talking to is making a statement or asking a question. Asking questions, giving orders, or making informational statements are things that people do in conversation, yet dealing with these kind of actions in dialogue—what we will call **dialog acts**— is something that the GUS-style frame-based dialog systems of Chapter 29 are completely incapable of.

In this chapter we describe the **dialog-state** architecture, also called the **belief-state** or **information-state** architecture. Like GUS systems, these agents fill slots, but they are also capable of understanding and generating such **dialog acts**, actions like asking a question, making a proposal, rejecting a suggestion, or acknowledging an utterance and they can incorporate this knowledge into a richer model of the state of the dialog at any point.

Like the GUS systems, the dialog-state architecture is based on filling in the slots of frames, and so dialog-state systems have an NLU component to determine the specific slots and fillers expressed in a user's sentence. Systems must additionally determine what dialog act the user was making, for example to track whether a user is asking a question. And the system must take into account the dialog context (what the system just said, and all the constraints the user has made in the past).

Furthermore, the dialog-state architecture has a different way of deciding what to say next than the GUS systems. Simple frame-based systems often just continuously ask questions corresponding to unfilled slots and then report back the results of some database query. But in natural dialogue users sometimes take the initiative, such as asking questions of the system; alternatively, the system may not understand what the user said, and may need to ask clarification questions. The system needs a **dialog policy** to decide what to say (when to answer the user's questions, when to instead ask the user a clarification question, make a suggestion, and so on).

Figure 30.1 shows a typical architecture for a dialog-state system. It has six components. As with the GUS-style frame-based systems, the speech recognition and understanding components extract meaning from the input, and the generation and TTS components map from meaning to speech.

The parts that are different than the simple GUS system are the **dialog state tracker** which maintains the current state of the dialog (which include the user's most recent dialog act, plus the entire set of slot-filler constraints the user has expressed so far) and the **dialog policy**, which decides what the system should do or say next.

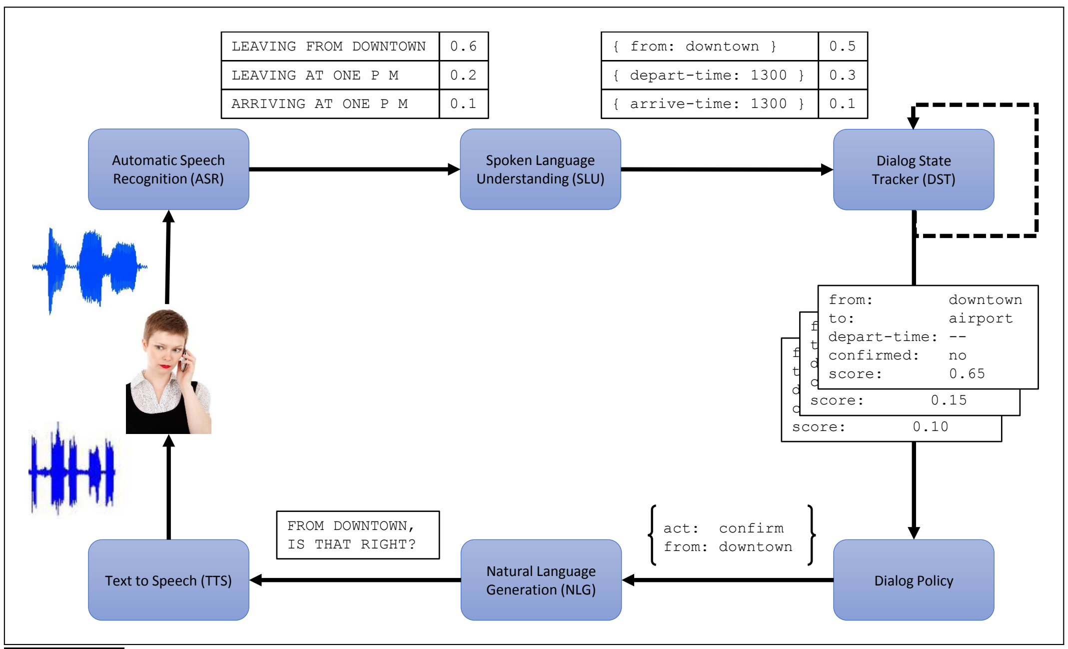

Figure 30.1 Architecture of a dialog-state system for task-oriented dialog from Williams et al. (2016).

As of the time of this writing, no commercial system uses a full dialog-state architecture, but some aspects of this architecture are beginning to appear in industrial systems, and there are a wide variety of these systems in research labs.

Let's turn first to a discussion of dialog acts.

30.1 Dialog Acts

A key insight into conversation—due originally to the philosopher Wittgenstein (1953) but worked out more fully by Austin (1962)—is that each utterance in a dialog is a kind of **action** being performed by the speaker. These actions are commonly called **speech acts**; here's one taxonomy consisting of 4 major classes (Bach

speech acts

and Harnish, 1979):

Constatives:	committing the speaker to something's being the case (*answering*, *claiming*, *confirming*, *denying*, *disagreeing*, *stating*)
Directives:	attempts by the speaker to get the addressee to do something (*advising*, *asking*, *forbidding*, *inviting*, *ordering*, *requesting*)
Commissives:	committing the speaker to some future course of action (*promising, planning, vowing, betting, opposing*)
Acknowledgments:	express the speaker's attitude regrading the hearer with respect to some social action (*apologizing*, *greeting*, *thanking*, *accepting an acknowledgment*)

A user ordering a dialog system to do something ('Turn up the music') is issuing a DIRECTIVE. A user asking a question to which the system is expected to answer is also issuing a DIRECTIVE: in a sense the user is commanding the system to answer ('What's the address of the second restaurant'). By contrast, a user stating a constraint ('I am flying on Tuesday') is issuing an ASSERTIVE. A user thanking the system is issuing an ACKNOWLEDGMENT. The dialog act expresses an important component of the intention of the speaker (or writer) in saying what they said.

While this idea of speech acts is powerful, modern systems expand these early taxonomies of speech acts to better describe actual conversations. This is because a dialog is not a series of unrelated independent speech acts, but rather a collective act performed by the speaker and the hearer. In performing this joint action the speaker and hearer must constantly establish **common ground** (Stalnaker, 1978), the set of things that are mutually believed by both speakers.

common ground

grounding

The need to achieve common ground means that the hearer must **ground** the speaker's utterances. To ground means to acknowledge, to make it clear that the hearer has understood the speaker's meaning and intention. People need closure or grounding for non-linguistic actions as well. For example, why does a well-designed elevator button light up when it's pressed? Because this indicates to the elevator traveler that she has successfully called the elevator. Clark (1996) phrases this need for closure as follows, after Norman (1988):

> **Principle of closure.** Agents performing an action require evidence, sufficient for current purposes, that they have succeeded in performing it.

Grounding is also important when the hearer needs to indicate that the speaker has *not* succeeded. If the hearer has problems in understanding, she must indicate these problems to the speaker, again so that mutual understanding can eventually be achieved.

Clark and Schaefer (1989) point out a continuum of methods the hearer B can use to ground the speaker A's utterance, ordered from weakest to strongest:

Continued attention:	B shows she is continuing to attend and therefore remains satisfied with A's presentation.
Next contribution:	B starts in on the next relevant contribution.
Acknowledgment:	B nods or says a continuer like *uh-huh*, *yeah*, or the like, or an **assessment** like *that's great*.
Demonstration:	B demonstrates all or part of what she has understood A to mean, for example, by **reformulating** (paraphrasing) A's utterance or by **collaborative completion** of A's utterance.
Display:	B displays verbatim all or part of A's presentation.

Let's look for examples of grounding in a conversation between a human travel agent and a human client in Fig. 30.2.

C_1:	... I need to travel in May.
A_1:	And, what day **in May** did you want to travel?
C_2:	OK uh I need to be there for a meeting that's from the 12th to the 15th.
A_2:	And you're flying into what city?
C_3:	Seattle.
A_3:	And what time would you like to leave Pittsburgh?
C_4:	Uh hmm I don't think there's many options for non-stop.
A_4:	Right. There's three non-stops today.
C_5:	What are they?
A_5:	The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
C_6:	OK I'll take the 5ish flight on the night before on the 11th.
A_6:	On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
C_7:	OK.

Figure 30.2 Part of a conversation between a travel agent (A) and client (C).

Utterance A_1 shows the strongest form of grounding, in which the hearer displays understanding by repeating verbatim part of the speaker's words: *in May*,

This particular fragment doesn't have an example of an *acknowledgment*, but there's an example in another fragment:

C:	He wants to fly from Boston to Baltimore
A:	**Uh huh**

backchannel
continuer

The word *uh-huh* here is a **backchannel**, also called a **continuer** or an **acknowledgment token**. A backchannel is a (short) optional utterance that acknowledges the content of the utterance of the other and that doesn't require an acknowledgment by the other (Yngve 1970, Jefferson 1984, Schegloff 1982, Ward and Tsukahara 2000).

The third grounding method is to start in on the relevant next contribution, for example in Fig. 30.2, where the speaker asks a question (A_2) and the hearer (C_3) answers it.

In a more subtle act of grounding, the speaker can combine this method with the previous one. For example, notice that whenever the client answers a question, the agent begins the next question with *And*. The *And* indicates to the client that the agent has successfully understood the answer to the last question:

Speech acts are important for practical dialog systems, which need to distinguish a statement from a directive, and which must distinguish (among the many kinds of directives) an order to do something from a question asking for information. Grounding is also crucial in dialog systems. Consider the unnaturalness of this example from Cohen et al. (2004):

(30.1) System: Did you want to review some more of your personal profile?
Caller: No.
System: What's next?

Without an acknowledgment, the caller doesn't know that the system has understand her 'No'. The use of *Okay* below adds grounding, making (30.2) a much more natural response than (30.1):

(30.2) System: Did you want to review some more of your personal profile?
Caller: No.

System: *Okay,* what's next?

Tag	Example
THANK	*Thanks*
GREET	*Hello Dan*
INTRODUCE	*It's me again*
BYE	*Alright bye*
REQUEST-COMMENT	*How does that look?*
SUGGEST	*from thirteenth through seventeenth June*
REJECT	*No Friday I'm booked all day*
ACCEPT	*Saturday sounds fine*
REQUEST-SUGGEST	*What is a good day of the week for you?*
INIT	*I wanted to make an appointment with you*
GIVE_REASON	*Because I have meetings all afternoon*
FEEDBACK	*Okay*
DELIBERATE	*Let me check my calendar here*
CONFIRM	*Okay, that would be wonderful*
CLARIFY	*Okay, do you mean Tuesday the 23rd?*
DIGRESS	*[we could meet for lunch] and eat lots of ice cream*
MOTIVATE	*We should go to visit our subsidiary in Munich*
GARBAGE	*Oops, I-*

Figure 30.3 The 18 high-level dialog acts for a meeting scheduling task, from the Verbmobil-1 system (Jekat et al., 1995).

dialog act

The ideas of speech acts and grounding are combined in a single kind of action called a **dialog act**, a tag which represents the interactive function of the sentence being tagged. Different types of dialog systems require labeling different kinds of acts, and so the tagset—defining what a dialog act is exactly— tends to be designed for particular tasks.

Figure 30.3 shows a domain-specific tagset for the task of two people scheduling meetings. It has tags specific to the domain of scheduling, such as SUGGEST, used for the proposal of a particular date to meet, and ACCEPT and REJECT, used for acceptance or rejection of a proposal for a date, but also tags that have more general function, like CLARIFY, used to request a user to clarify an ambiguous proposal.

Tag	Sys	User	Description
HELLO($a=x,b=y,\ldots$)	✓	✓	Open a dialog and give info $a=x,b=y,\ldots$
INFORM($a=x,b=y,\ldots$)	✓	✓	Give info $a=x,b=y,\ldots$
REQUEST($a,b=x,\ldots$)	✓	✓	Request value for a given $b=x,\ldots$
REQALTS($a=x,\ldots$)	χ	✓	Request alternative with $a=x,\ldots$
CONFIRM($a=x,b=y,\ldots$)	✓	✓	Explicitly confirm $a=x,b=y,\ldots$
CONFREQ($a=x,\ldots,d$)	✓	χ	Implicitly confirm $a=x,\ldots$ and request value of d
SELECT($a=x,a=y$)	✓	χ	Implicitly confirm $a=x,\ldots$ and request value of d
AFFIRM($a=x,b=y,\ldots$)	✓	✓	Affirm and give further info $a=x,b=y,\ldots$
NEGATE($a=x$)	χ	✓	Negate and give corrected value $a=x$
DENY($a=x$)	χ	✓	Deny that $a=x$
BYE()	✓	✓	Close a dialog

Figure 30.4 Dialogue acts used by the HIS restaurant recommendation system of Young et al. (2010). The Sys and User columns indicate which acts are valid as system outputs and user inputs, respectively.

Figure 30.4 shows a tagset for a restaurant recommendation system, and Fig. 30.5 shows these tags labeling a sample dialog from the HIS system (Young et al., 2010). This example also shows the content of each dialog acts, which are the slot fillers being communicated.

	Utterance	Dialogue act
U:	Hi, I am looking for somewhere to eat.	`hello(task = find,type=restaurant)`
S:	You are looking for a restaurant. What type of food do you like?	`confreq(type = restaurant, food)`
U:	I'd like an Italian somewhere near the museum.	`inform(food = Italian, near=museum)`
S:	Roma is a nice Italian restaurant near the museum.	`inform(name = "Roma", type = restaurant, food = Italian, near = museum)`
U:	Is it reasonably priced?	`confirm(pricerange = moderate)`
S:	Yes, Roma is in the moderate price range.	`affirm(name = "Roma", pricerange = moderate)`
U:	What is the phone number?	`request(phone)`
S:	The number of Roma is 385456.	`inform(name = "Roma", phone = "385456")`
U:	Ok, thank you goodbye.	`bye()`

Figure 30.5 A sample dialog from the HIS System of Young et al. (2010) using the dialog acts in Fig. 30.4.

conversational analysis

adjacency pair

Dialog acts don't just appear discretely and independently; conversations have structure, and dialogue acts reflect some of that structure. One aspect of this structure comes from the field of **conversational analysis** or CA (Sacks et al., 1974) which focuses on interactional properties of human conversation. CA defines **adjacency pairs** (Schegloff, 1968) as a pairing of two dialog acts, like QUESTIONS and ANSWERS, PROPOSAL and ACCEPTANCE (or REJECTION), COMPLIMENTS and DOWNPLAYERS, GREETING and GREETING.

side sequence

subdialogue

The structure, composed of a **first pair part** and a**second pair part**, can help dialog-state models decide what actions to take. However, dialog acts aren't always followed immediately by their second pair part. The two parts can be separated by a **side sequence** (Jefferson 1972, Schegloff 1972). One very common side sequence in dialog systems is the **clarification question**, which can form a **subdialogue** between a REQUEST and a RESPONSE as in the following example caused by speech recognition errors:

User: What do you have going to UNKNOWN_WORD on the 5th?
  System: Let's see, going where on the 5th?
  User: Going to Hong Kong.
System: OK, here are some flights...

pre-sequence

Another kind of dialogue structure is the **pre-sequence**, like the following example where a user starts with a question about the system's capabilities ("Can you make train reservations") before making a request.

User: Can you make train reservations?
System: Yes I can.
User: Great, I'd like to reserve a seat on the 4pm train to New York.

A dialog-state model must be able to both recognize these kinds of structures and make use of them in interacting with users.

30.2 Dialog State: Interpreting Dialogue Acts

The job of the dialog-state tracker is to determine both the current state of the frame (the fillers of each slot), as well as the user's most recent dialog act. Note that the dialog-state includes more than just the slot-fillers expressed in the current sentence; it includes the entire state of the frame at this point, summarizing all of the user's constraints. The following example from Mrkšić et al. (2017) shows the required output of the dialog state tracker after each turn:

User:	I'm looking for a cheaper restaurant
	`inform(price=cheap)`
System:	Sure. What kind - and where?
User:	Thai food, somewhere downtown
	`inform(price=cheap, food=Thai, area=centre)`
System:	The House serves cheap Thai food
User:	Where is it?
	`inform(price=cheap, food=Thai, area=centre); request(address)`
System:	The House is at 106 Regent Street

How can we interpret a dialog act, deciding whether a given input is a QUESTION, a STATEMENT, or a SUGGEST (directive)? Surface syntax seems like a useful cue, since yes-no questions in English have **aux-inversion** (the auxiliary verb precedes the subject), statements have declarative syntax (no aux-inversion), and commands have no syntactic subject:

(30.3)	YES-NO QUESTION	Will breakfast be served on USAir 1557?
	STATEMENT	I don't care about lunch.
	COMMAND	Show me flights from Milwaukee to Orlando.

Alas, the mapping from surface form to dialog act is complex. For example, the following utterance looks grammatically like a YES-NO QUESTION meaning something like *Are you capable of giving me a list of... ?*:

(30.4) Can you give me a list of the flights from Atlanta to Boston?

In fact, however, this person was not interested in whether the system was *capable* of giving a list; this utterance was a polite form of a REQUEST, meaning something like *Please give me a list of....* What looks on the surface like a QUESTION can really be a REQUEST.

Conversely, what looks on the surface like a STATEMENT can really be a QUESTION. The very common CHECK question (Carletta et al. 1997, Labov and Fanshel 1977) asks an interlocutor to confirm something that she has privileged knowledge about. CHECKS have declarative surface form:

A	OPEN-OPTION	I was wanting to make some arrangements for a trip that I'm going to be taking uh to LA uh beginning of the week after next.
B	HOLD	OK uh let me pull up your profile and I'll be right with you here. [pause]
B	CHECK	**And you said you wanted to travel next week?**
A	ACCEPT	Uh yes.

indirect speech act

Utterances that use a surface statement to ask a question or a surface question to issue a request are called **indirect speech acts**. These indirect speech acts have a rich literature in philosophy, but viewed from the perspective of dialog understanding, indirect speech acts are merely one instance of the more general problem of determining the dialog act function of a sentence.

prosody
intonation

Many features can help in this task. To give just one example, in spoken-language systems, **prosody** or **intonation** (Chapter **??**) is a helpful cue. Prosody or intonation is the name for a particular set of phonological aspects of the speech signal the **tune** and other changes in the pitch (which can be extracted from the fundamental frequency F0) the **accent**, stress, or loudness (which can be extracted from energy), and the changes in duration and **rate of speech**. So, for example, a rise in pitch at the end of the utterance is a good cue for a YES-NO QUESTION, while declarative utterances (like STATEMENTS) have **final lowering**: a drop in F0 at the end of the utterance.

final lowering

30.2.1 Sketching an algorithm for dialog act interpretation

Since dialog acts places some constraints on the slots and values, the tasks of dialog-act detection and slot-filling are often performed jointly. Consider the task of determining that

```
I'd like Cantonese food near the Mission District
```

has the structure

```
inform(food=cantonese,area=mission)).
```

The joint dialog act interpretation/slot filling algorithm generally begins with a first pass classifier to decide on the dialog act for the sentence. In the case of the example above, this classifier would choosing `inform` from among the set of possible dialog acts in the tag set for this particular task. Dialog act interpretation is generally modeled as a supervised classification task, trained on a corpus in which each utterance is hand-labeled for its dialog act, and relying on a wide variety of features, including unigrams and bigrams (*show me* is a good cue for a REQUEST, *are there* for a QUESTION), parse features, punctuation, dialog context, and the prosodic features described above.

A second pass classifier might use any of the algorithms for slot-filler extraction discussed in Section **??** of Chapter 29, such as CRF or RNN-based IOB tagging. Alternatively, a multinominal classifier can be used to choose between all possible slot-value pairs, again using any of the feature functions defined in Chapter 29. This is possible since the domain ontology for the system is fixed, so there is a finite number of slot-value pairs.

Both classifiers can be built from any standard multinominal classifier (logistic regression, SVM), using the various features described above, or, if sufficient training data is available, can be built with end-to-end neural models.

30.2.2 A special case: detecting correction acts

Some dialog acts are important because of their implications for dialog control. If a dialog system misrecognizes or misunderstands an utterance, the user will generally correct the error by repeating or reformulating the utterance. Detecting these **user correction acts** is therefore quite important. Ironically, it turns out that corrections are actually *harder* to recognize than normal sentences! In fat, corrections in one early dialog system (the TOOT system) had double the ASR word error rate of non-corrections Swerts et al. (2000)! One reason for this is that speakers sometimes use a specific prosodic style for corrections called **hyperarticulation**, in which the

user correction acts

hyperarticulation

utterance contains some exaggerated energy, duration, or F0 contours, such as *I said BAL-TI-MORE, not Boston* (Wade et al. 1992, Levow 1998, Hirschberg et al. 2001). Even when they are not hyperarticulating, users who are frustrated seem to speak in a way that is harder for speech recognizers (Goldberg et al., 2003).

What are the characteristics of these corrections? User corrections tend to be either exact repetitions or repetitions with one or more words omitted, although they may also be paraphrases of the original utterance. (Swerts et al., 2000). Detecting these reformulations or correction acts can be done by any classifier; some standard features used for this task are shown below (Levow 1998, Litman et al. 1999, Hirschberg et al. 2001, Bulyko et al. 2005, Awadallah et al. 2015):

lexical features	words like "no", "correction", "I don't", or even swear words, utterance length
semantic features	overlap between the candidate correction act and the user's prior utterance (computed by word overlap or via cosines over embedding vectors)
phonetic features	phonetic overlap between the candidate correction act and the user's prior utterance (i.e. "WhatsApp" may be incorrectly recognized as "What's up")
prosodic features	hyperarticulation, increases in F0 range, pause duration, and word duration, generally normalized by the values for previous sentences
ASR features	ASR confidence, language model probability

30.3 Dialogue Policy

dialog policy

The goal of the **dialog policy** is to decide what action the system should take next, that is, what dialog act to generate. We begin in the next section by introducing one specific dialog policy decision, relating to confirmation: how we confirm to the user what we think she said. We then sketch a basic policy algorithm that could apply to all decisions. Finally, once a speech act has been generated, the natural language generation component needs to generate the text of a response to the user.

30.3.1 Generating Dialogue Acts: Confirmation and Rejection

Modern dialog systems often make mistakes. It is therefore important for dialog systems to make sure that they have achieved the correct interpretation of the user's input. This is generally done by two methods: **confirming** understandings with the user and **rejecting** utterances that the system is likely to have misunderstood.

explicit confirmation

Various strategies can be employed for confirmation with the user. When using the **explicit confirmation** strategy, a system asks the user a direct question to confirm the system's understanding, like the two examples below in which the system asks a (boldface) yes-no confirmation questions:

S:	Which city do you want to leave from?
U:	Baltimore.
S:	**Do you want to leave from Baltimore?**
U:	Yes.
U:	I'd like to fly from Denver Colorado to New York City on September twenty first in the morning on United Airlines
S:	**Let's see then. I have you going from Denver Colorado to New York on September twenty first. Is that correct?**
U:	Yes

implicit confirmation

When using the **implicit confirmation** strategy, a system instead uses the *demonstration* or *display* grounding strategies described above, repeating back the system's understanding as part of asking the next question, as in the two examples below:

U:	I want to travel to Berlin
S:	**When do you want to travel to Berlin?**
U2:	Hi I'd like to fly to Seattle Tuesday Morning
A3:	**Traveling to Seattle on Tuesday, August eleventh in the morning.** Your full name?

Explicit and implicit confirmation have complementary strengths. Explicit confirmation makes it easier for users to correct the system's misrecognitions since a user can just answer "no" to the confirmation question. But explicit confirmation is awkward and increases the length of the conversation (Danieli and Gerbino 1995, Walker et al. 1998). The explicit confirmation dialog fragments above sound non-natural and definitely non-human; implicit confirmation is much more conversationally natural.

rejection

Confirmation is just one kind of conversational action by which a system can express lack of understanding. Another option is **rejection**, in which a system gives the user a prompt like *I'm sorry, I didn't understand that*.

progressive prompting

Sometimes utterances are rejected multiple times. This might mean that the user is using language that the system is unable to follow. Thus, when an utterance is rejected, systems often follow a strategy of **progressive prompting** or **escalating detail** (Yankelovich et al. 1995, Weinschenk and Barker 2000), as in this example from Cohen et al. (2004):

System:	When would you like to leave?
Caller:	Well, um, I need to be in New York in time for the first World Series game.
System:	<reject>. Sorry, I didn't get that. Please say the month and day you'd like to leave.
Caller:	I wanna go on October fifteenth.

In this example, instead of just repeating "When would you like to leave?", the rejection prompt gives the caller more guidance about how to formulate an utterance the system will understand. These *you-can-say* help messages are important in helping improve systems' understanding performance (Bohus and Rudnicky, 2005). If the caller's utterance gets rejected yet again, the prompt can reflect this ("I *still* didn't get that"), and give the caller even more guidance.

rapid reprompting

An alternative strategy for error handling is **rapid reprompting**, in which the system rejects an utterance just by saying "I'm sorry?" or "What was that?" Only if the caller's utterance is rejected a second time does the system start applying progressive prompting. Cohen et al. (2004) summarize experiments showing that users greatly prefer rapid reprompting as a first-level error prompt.

Various factors can be used as features to the dialog policy in deciding whether to use explicit confirmation, implicit confirmation, or rejection. For example, the **confidence** that the ASR system assigns to an utterance can be used by explicitly confirming low-confidence sentences. Recall from page **??** that confidence is a metric that the speech recognizer can assign to its transcription of a sentence to indicate how confident it is in that transcription. Confidence is often computed from the acoustic log-likelihood of the utterance (greater probability means higher confidence), but prosodic features can also be used in confidence prediction. For example, utterances with large F0 excursions or longer durations, or those preceded by longer pauses, are likely to be misrecognized (Litman et al., 2000).

Another common feature in confirmation is the **cost** of making an error. For example, explicit confirmation is common before a flight is actually booked or money in an account is moved. Systems might have a four-tiered level of confidence with three thresholds α, β, and γ:

$< \alpha$	low confidence	reject
$\geq \alpha$	above the threshold	confirm explicitly
$\geq \beta$	high confidence	confirm implictly
$\geq \gamma$	very high confidence	don't confirm at all

30.4 A simple policy based on local context

The goal of the dialog policy at turn i in the conversation is to predict which action A_i to take, based on the entire dialog state. The state could mean the entire sequence of dialog acts from the system (A) and from the user (U), in which case the task would be to compute:

$$\hat{A}_i = \underset{A_i \in A}{\operatorname{argmax}} P(A_i | (A_1, U_1, ..., A_{i-1}, U_{i-1}) \tag{30.5}$$

We can simplify this by maintaining as the dialog state mainly just the set of slot-fillers that the user has expressed, collapsing across the many different conversational paths that could lead to the same set of filled slots.

Such a policy might then just condition on the current state of the frame Frame_i (which slots are filled and with what) and the last turn by the system and user:

$$\hat{A}_i = \underset{A_i \in A}{\operatorname{argmax}} P(A_i | \text{Frame}_{i-1}, A_{i-1}, U_{i-1}) \tag{30.6}$$

Given a large enough corpus of conversations, these probabilities can be estimated by a classifier. Getting such enormous amounts of data can be difficult, and often involves building user simulators to generate artificial conversations to train on.

30.5 Natural language generation in the dialog-state model

content planning
sentence realization

Once a dialog act has been decided, we need to generate the text of the response to the user. The task of natural language generation (NLG) in the information-state architecture is often modeled in two stages, **content planning** (what to say), and **sentence realization** (how to say it).

Here we'll assume content planning has been done by the dialog policy, which has chosen the dialog act to generate, and perhaps also chosen some some additional attributes (slots and values) that the planner wants to implicitly confirm to the user. Fig. 30.6 shows a sample input structure from the policy/content planner, and one example of a resulting sentence that the sentence realizer could generate from this structure.

Let's walk through the sentence realization stage for the example in Fig. 30.6, which comes from the classic information state statistical NLG system of Oh and

```
{
  act query
  content depart_time
  depart_date {
        year 2000
        month 10
        day 5
       }
  depart_airport BOS
}
=> What time on October fifth would you like to leave Boston?
```

Figure 30.6 An input frame to NLG and a resulting output sentence, in the Communicator system of Oh and Rudnicky (2000).

query arrive_city	hotel hotel_chain	inform flight_earlier
query arrive_time	hotel hotel_info	inform flight_earliest
query confirm	hotel need_car	inform flight_later
query depart_date	hotel need_hotel	inform flight_latest
query depart_time	hotel where	inform flight_returning
query pay_by_card	inform airport	inform not_avail
query preferred_airport	inform confirm_utterance	inform num_flights
query return_date	inform epilogue	inform price
query return_time	inform flight	other
hotel car_info	inform flight_another	

Figure 30.7 Dialog acts in the CMU communicator system of Oh and Rudnicky (2000).

Rudnicky (2000), part of the CMU Communicator travel planning dialog system. Notice first that the policy has decided to generate the dialog act QUERY with the argument DEPART_TIME. Fig. 30.7 lists the dialog acts in the Oh and Rudnicky (2000) system, each of which combines an act with a potential argument. The input frame in Fig. 30.6 also specifies some additional filled slots that should be included in the sentence to the user (depart_airport BOS, and the depart_date).

delexicalized The sentence realizer acts in two steps. It will first generate a **delexicalized** string like:

What time on [depart_date] would you like to leave [depart_airport]?

Delexicalization is the process of replacing specific words with a generic representation of their slot types. A delexicalized sentence is much easier to generate since we can train on many different source sentences from different specific dates and airports. Then once we've generating the delexicalized string, we can simply use **relexicalize** the input frame from the content planner to **relexicalize** (fill in the exact departure date and airport).

To generate the delexicalized sentences, the sentence realizer uses a large corpus of human-human travel dialogs that were labeled with the dialog acts from Fig. 30.7 and the slots expressed in each turn, like the following:

QUERY DEPART_TIME	And what time would you like to leave [$_{\text{depart\_city}}$ **Pittsburgh**]?
QUERY ARRIVE_CITY	And you're flying into what city?
QUERY ARRIVE_TIME	What time on [$_{\text{arrive\_date}}$ **May 5**]?
INFORM FLIGHT	The flight departs [$_{\text{depart\_airport}}$ **PGH**] at [$_{\text{depart\_time}}$ **10 am**] and arrives [$_{\text{arrive\_city}}$ **Seattle**] at [$_{\text{arrive\_time}}$ **12:05 their time**].

This corpus is then delexicalized, and divided up into separate corpora for each dialog act. Thus the delexicalized corpus for one dialog act, QUERY DEPART_TIME might be trained on examples like:

And what time would you like to leave depart_city?
When would you like to leave depart_city?
When would you like to leave?
What time do you want to leave on depart_date?
OK, on depart_date, what time do you want to leave?

A distinct N-gram grammar is then trained for each dialog act. Now, given the dialog act QUERY DEPART_TIME, the system samples random sentences from this language model. Recall from the the "Shannon" exercise of **??** that this works (assuming a bigram LM) by first selecting a bigram ($< s >, < w >$) according to its bigram probability in the language model, then drawing a bigram starting with $< w >$ according to its bigram probability, and so on until a full sentence is generated. The probability of each successive word w_i being generated from utterance class u is thus

$$P(w_i) = P(w_i|w_{i-1}, w_{i-2}, ..., w_{i-(n-1)}, u) \tag{30.7}$$

Each of these randomly sampled sentences is then assigned a score based on heuristic rules that penalize sentences that are too short or too long, repeat slots, or lack some of the required slots from the input frame (in this case, depart_airport and depart_date). The best scoring sentence is then chosen. Let's suppose in this case we produce the following (delexicalized) sentence:

What time on depart_date would you like to leave depart_airport?

This sentence is then relexicalized from the true values in the input frame, resulting in the final sentence:

What time on October fifth would you like to leave Boston?

More recent work has replaced the simplistic N-gram part of the generator with neural models, which similarly learn to map from an input frame to a resulting sentence (Wen et al. 2015a, Wen et al. 2015b).

clarification questions

It's also possible to design NLG algorithms that are specific to a particular dialog act. For example, consider the task of generating **clarification questions**, in cases where the speech recognition fails to understand some part of the user's utterance. While it is possible to use the generic dialog act REJECT ("Please repeat", or "I don't understand what you said"), studies of human conversations show that humans instead use targeted clarification questions that reprise elements of the misunderstanding (Purver 2004, Ginzburg and Sag 2000, Stoyanchev et al. 2013).

For example, in the following hypothetical example the system reprises the words "going" and "on the 5th" to make it clear which aspect of the user's turn the system needs to be clarified:

User:	What do you have going to UNKNOWN_WORD on the 5th?
System:	Going where on the 5th?

Targeted clarification questions can be created by rules (such as replacing "going to UNKNOWN_WORD" with "going where") or by building classifiers to guess which slots might have been misrecognized in the sentence (Chu-Carroll and Carpenter 1999, Stoyanchev et al. 2014, Stoyanchev and Johnston 2015).

30.6 Advanced: Markov Decision Processes

The policy we described in Section 30.4, deciding what actions the system should take based just on the current filled slots and the users last utterance, has a problem: it looks only at the past of the dialog, completely ignoring whether the action we take is likely to lead to a successful outcome (a correctly booked flight or filled-in calendar).

But we can't know whether the outcome is successful until long after the current utterance we are trying to plan. Reinforcement learning is the branch of machine learning that deals with models that learn to maximize future rewards.

This is an extremely active area of research, so we give here just the simplest intuition for this direction, based on an oversimplified model of dialog as a **Markov decision process**.

Markov decision process
MDP

A Markov decision process or **MDP** is characterized by a set of **states** S an agent can be in, a set of **actions** A the agent can take, and a **reward** $r(a,s)$ that the agent receives for taking an action in a state. Given these factors, we can compute a **policy** π that specifies which action a the agent should take when in a given state s so as to receive the best reward.

To understand each of these components, we need to look at a tutorial example in which the state space is extremely reduced. Let's look at a trivial pedagogical frame-and-slot example from Levin et al. (2000), a "Day-and-Month" dialog system whose goal is to get correct values of day and month for a two-slot frame through the shortest possible interaction with the user.

In principle, a state of an MDP could include any possible information about the dialog, such as the complete dialog history so far. Using such a rich model of state would make the number of possible states extraordinarily large. So a model of state is usually chosen that encodes a much more limited set of information, such as the values of the slots in the current frame, the most recent question asked to the user, the user's most recent answer, the ASR confidence, and so on. For the Day-and-Month example, let's represent the state of the system as the values of the two slots *day* and *month*. There are 411 states (366 states with a day and month (counting leap year), 12 states with a month but no day ($d = 0$, $m = 1,2,...,12$), 31 states with a day but no month ($m = 0$, $d = 1,2,...,31$), and a special initial state s_i and final state s_f.

Actions of an MDP dialog system might include generating particular speech acts, or performing a database query to find out information. For the Day-and-Month example, Levin et al. (2000) propose the following actions:

- a_d: a question asking for the day
- a_m: a question asking for the month
- $a_d m$: a question asking for both the day and the month
- a_f: a final action submitting the form and terminating the dialog

Since the goal of the system is to get the correct answer with the shortest interaction, one possible reward function for the system would integrate three terms:

$$R = -(w_i n_i + w_e n_e + w_f n_f) \tag{30.8}$$

The term n_i is the number of interactions with the user, n_e is the number of errors, n_f is the number of slots that are filled (0, 1, or 2), and the ws are weights.

Finally, a dialog policy π specifies which actions to apply in which state. Consider two possible policies: (1) asking for day and month separately, and (2) asking for them together. These might generate the two dialogs shown in Fig. 30.8.

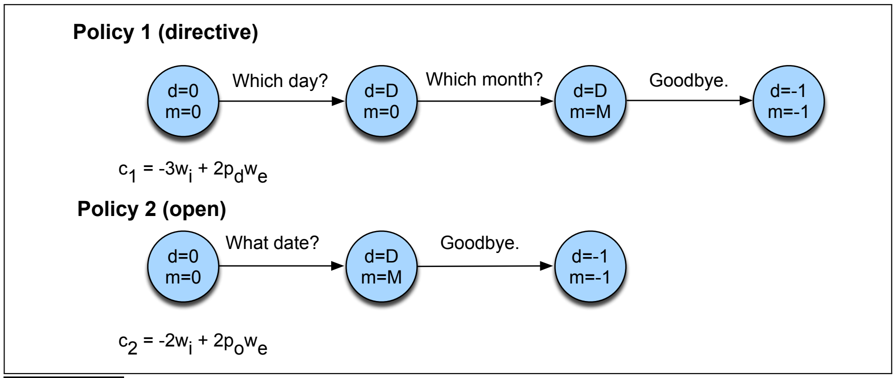

Figure 30.8 Two policies for getting a month and a day. After Levin et al. (2000).

In policy 1, the action specified for the no-date/no-month state is to ask for a day, and the action specified for any of the 31 states where we have a day but not a month is to ask for a month. In policy 2, the action specified for the no-date/no-month state is to ask an open-ended question (*Which date*) to get both a day and a month. The two policies have different advantages; an open prompt can lead to shorter dialogs but is likely to cause more errors, whereas a directive prompt is slower but less error-prone. Thus, the optimal policy depends on the values of the weights w and also on the error rates of the ASR component. Let's call p_d the probability of the recognizer making an error interpreting a month or a day value after a directive prompt. The (presumably higher) probability of error interpreting a month or day value after an open prompt we'll call p_o. The reward for the first dialog in Fig. 30.8 is thus $-3 \times w_i + 2 \times p_d \times w_e$. The reward for the second dialog in Fig. 30.8 is $-2 \times w_i + 2 \times p_o \times w_e$. The directive prompt policy, policy 1, is thus better than policy 2 when the improved error rate justifies the longer interaction, that is, when $p_d - p_o > \frac{w_i}{2w_e}$.

In the example we've seen so far, there were only two possible actions, and hence only a tiny number of possible policies. In general, the number of possible actions, states, and policies is quite large, and so the problem of finding the optimal policy π^* is much harder.

Markov decision theory together with classical reinforcement learning gives us a way to think about this problem. First, generalizing from Fig. 30.8, we can think of any particular dialog as a trajectory in state space:

$$s_1 \rightarrow_{a1,r1} s_2 \rightarrow_{a2,r2} s_3 \rightarrow_{a3,r3} \cdots \tag{30.9}$$

The best policy π^* is the one with the greatest expected reward over all trajectories. What is the expected reward for a given state sequence? The most common way to assign utilities or rewards to sequences is to use **discounted rewards**. Here we compute the expected cumulative reward Q of a sequence as a discounted sum of the utilities of the individual states:

discounted reward

$$Q([s_0, a_0, s_1, a_1, s_2, a_2 \cdots]) = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots, \tag{30.10}$$

The discount factor γ is a number between 0 and 1. This makes the agent care more about current rewards than future rewards; the more future a reward, the more discounted its value.

Given this model, it is possible to show that the expected cumulative reward $Q(s,a)$ for taking a particular action from a particular state is the following recursive equation called the **Bellman equation**:

Bellman equation

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \tag{30.11}$$

What the Bellman equation says is that the expected cumulative reward for a given state/action pair is the immediate reward for the current state plus the expected discounted utility of all possible next states s', weighted by the probability of moving to that state s', and assuming that once there we take the optimal action a'.

Equation 30.11 makes use of two parameters. We need a model of $P(s'|s,a)$, that is, how likely a given state/action pair (s,a) is to lead to a new state s'. And we also need a good estimate of $R(s,a)$. If we had lots of labeled training data, we could simply compute both of these from labeled counts. For example, with labeled dialogs, to estimate $P(s'|s,a)$ we could simply count how many times we were in a given state s, and out of that how many times we took action a to get to state s'. Similarly, if we had a hand-labeled reward for each dialog, we could build a model of $R(s,a)$.

value iteration

Given these parameters, there is an iterative algorithm for solving the Bellman equation and determining proper Q values, the **value iteration** algorithm (Sutton and Barto 1998, Bellman 1957). See Russell and Norvig (2002) for the details of the algorithm.

How do we get enough labeled training data to set these parameters? This is especially worrisome since in real problems the number of states s is extremely large. The most common method is to build a simulated user. The user interacts with the system millions of times, and the system learns the state transition and reward probabilities from this corpus. For example Levin et al. (2000) build a generative stochastic model that given the system's current state and actions, produced a frame-slot representation of a user response; the parameters of the simulated user were estimated from a corpus of ATIS dialogs.

The MDP is only useful in small toy examples and is not used in practical dialog systems. A more powerful model, the partially observable Markov decision process, or POMDP, adds extra latent variables to represent our uncertainty about the true state of the dialog. Both MDPs and POMDPs, however, have problems due to computational complexity and due to their reliance on simulations that don't reflect true user behavior.

Recent research has therefore focused on ways to build real task-based systems that nonetheless make use of this reinforcement learning intuition, often by adding reinforcement learning to deep neural networks. This is an exciting new area of research, but a standard paradigm has yet to emerge.

30.7 Summary

- In dialog, speaking is a kind of action; these acts are referred to as speech acts. Speakers also attempt to achieve **common ground** by acknowledging that they have understand each other. The **dialog act** combines the intuition of speech acts and grounding acts.
- The **dialog-state** or information-state architecture augments the frame-and-slot state architecture by keeping track of user's dialog acts and includes a **policy** for generating its own dialog acts in return.
- Policies based on reinforcement learning architecture like the MDP and POMDP

offer ways for future dialog reward to be propagated back to influence policy earlier in the dialog manager.

Bibliographical and Historical Notes

The idea that utterances in a conversation are a kind of **action** being performed by the speaker was due originally to the philosopher Wittgenstein (1953) but worked out more fully by Austin (1962) and his student John Searle. Various sets of speech acts have been defined over the years, and a rich linguistic and philosophical literature developed, especially focused on explaining the use of indirect speech acts.

The idea of dialog acts draws also from a number of other sources, including the ideas of adjacency pairs, pre-sequences, and other aspects of the international properties of human conversation developed in the field of conversation analysis (see Levinson (1983) for an introduction to the field).

This idea that acts set up strong local dialogue expectations was also prefigured by Firth (1935, p. 70), in a famous quotation:

> *Most of the give-and-take of conversation in our everyday life is stereotyped and very narrowly conditioned by our particular type of culture. It is a sort of roughly prescribed social ritual, in which you generally say what the other fellow expects you, one way or the other, to say.*

Another important research thread modeled dialog as a kind of collaborative behavior, including the ideas of common ground (Clark and Marshall, 1981), reference as a collaborative process (Clark and Wilkes-Gibbs, 1986), joint intention (Levesque et al., 1990), and shared plans (Grosz and Sidner, 1980).

The information state model of dialogue was also strongly informed by analytic work on the linguistic properties of dialog acts and on methods for their detection (Sag and Liberman 1975, Hinkelman and Allen 1989, Nagata and Morimoto 1994, Goodwin 1996, Chu-Carroll 1998, Shriberg et al. 1998, Stolcke et al. 2000, Gravano et al. 2012).

Two important lines of research focused on the computational properties of conversational structure. One line, first suggested at by Bruce (1975), suggested that since speech acts are actions, they should be planned like other actions, and drew on the AI planning literature (Fikes and Nilsson, 1971). An agent seeking to find out some information can come up with the plan of asking the interlocutor for the information. An agent hearing an utterance can interpret a speech act by running the planner "in reverse", using inference rules to infer from what the interlocutor said what the plan might have been. Plan-based models of dialogue are referred to as **BDI** models because such planners model the **beliefs**, **desires**, and **intentions** (BDI) of the agent and interlocutor. BDI models of dialogue were first introduced by Allen, Cohen, Perrault, and their colleagues in a number of influential papers showing how speech acts could be generated (Cohen and Perrault, 1979) and interpreted (Perrault and Allen 1980, Allen and Perrault 1980). At the same time, Wilensky (1983) introduced plan-based models of understanding as part of the task of interpreting stories.

BDI

Another influential line of research focused on modeling the hierarchical structure of dialog. Grosz's pioneering (1977) dissertation first showed that "task-oriented dialogs have a structure that closely parallels the structure of the task being performed" (p. 27), leading to her work with Sidner and others showing how to use similar notions of intention and plans to model discourse structure and coherence in dialogue. See, e.g., Lochbaum et al. (2000) for a summary of the role of intentional structure in dialog.

The idea of applying reinforcement learning to dialogue first came out of AT&T and Bell Laboratories around the turn of the century with work on MDP dialogue systems (Walker 2000, Levin et al. 2000, Singh et al. 2002) and work on cue phrases, prosody, and rejection and confirmation. Reinforcement learning research turned quickly to the more sophisticated POMDP models (Roy et al. 2000, Lemon et al. 2006, Williams and Young 2007) applied to small slot-filling dialogue tasks.

More recent work has applied deep learning to many components of dialogue systems.

Allen, J. and Perrault, C. R. (1980). Analyzing intention in utterances. *Artificial Intelligence*, *15*, 143–178.

Austin, J. L. (1962). *How to Do Things with Words*. Harvard University Press.

Awadallah, A. H., Kulkarni, R. G., Ozertem, U., and Jones, R. (2015). Charaterizing and predicting voice query reformulation. In *CIKM-15*.

Bach, K. and Harnish, R. (1979). *Linguistic communication and speech acts*. MIT Press.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Bohus, D. and Rudnicky, A. I. (2005). Sorry, I didn't catch that! — An investigation of non-understanding errors and recovery strategies. In *Proceedings of SIGDIAL*, Lisbon, Portugal.

Bruce, B. C. (1975). Generation as a social action. In *Proceedings of TINLAP-1 (Theoretical Issues in Natural Language Processing)*, pp. 64–67. Association for Computational Linguistics.

Bulyko, I., Kirchhoff, K., Ostendorf, M., and Goldberg, J. (2005). Error-sensitive response generation in a spoken language dialogue system. *Speech Communication*, *45*(3), 271–288.

Carletta, J., Isard, A., Isard, S., Kowtko, J. C., Doherty-Sneddon, G., and Anderson, A. H. (1997). The reliability of a dialogue structure coding scheme. *Computational Linguistics*, *23*(1), 13–32.

Chu-Carroll, J. (1998). A statistical model for discourse act recognition in dialogue interactions. In Chu-Carroll, J. and Green, N. (Eds.), *Applying Machine Learning to Discourse Processing. Papers from the 1998 AAAI Spring Symposium.* Tech. rep. SS-98-01, pp. 12–17. AAAI Press.

Chu-Carroll, J. and Carpenter, B. (1999). Vector-based natural language call routing. *Computational Linguistics*, *25*(3), 361–388.

Clark, H. H. (1996). *Using Language*. Cambridge University Press.

Clark, H. H. and Marshall, C. (1981). Definite reference and mutual knowledge. In Joshi, A. K., Webber, B. L., and Sag, I. A. (Eds.), *Elements of Discourse Understanding*, pp. 10–63. Cambridge.

Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, *13*, 259–294.

Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, *22*, 1–39.

Cohen, M. H., Giangola, J. P., and Balogh, J. (2004). *Voice User Interface Design*. Addison-Wesley.

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

Danieli, M. and Gerbino, E. (1995). Metrics for evaluating dialogue strategies in a spoken language system. In *Proceedings of the 1995 AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, Stanford, CA, pp. 34–39. AAAI Press.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Firth, J. R. (1935). The technique of semantics. *Transactions of the philological society*, *34*(1), 36–73.

Ginzburg, J. and Sag, I. (2000). *Interrogative Investigations: the Form, Meaning and Use of English Interrogatives*. CSLI.

Goldberg, J., Ostendorf, M., and Kirchhoff, K. (2003). The impact of response wording in error correction subdialogs. In *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*.

Goodwin, C. (1996). Transparent vision. In Ochs, E., Schegloff, E. A., and Thompson, S. A. (Eds.), *Interaction and Grammar*, pp. 370–404. Cambridge University Press.

Gravano, A., Hirschberg, J., and Beňuš, Š. (2012). Affirmative cue words in task-oriented dialogue. *Computational Linguistics*, *38*(1), 1–39.

Grosz, B. J. (1977). *The Representation and Use of Focus in Dialogue Understanding*. Ph.D. thesis, University of California, Berkeley.

Grosz, B. J. and Sidner, C. L. (1980). Plans for discourse. In Cohen, P. R., Morgan, J., and Pollack, M. E. (Eds.), *Intentions in Communication*, pp. 417–444. MIT Press.

Hinkelman, E. A. and Allen, J. (1989). Two constraints on speech act ambiguity. In *ACL-89*, Vancouver, Canada, pp. 212–219.

Hirschberg, J., Litman, D. J., and Swerts, M. (2001). Identifying user corrections automatically in spoken dialogue systems. In *NAACL 2001*.

Jefferson, G. (1972). Side sequences. In Sudnow, D. (Ed.), *Studies in social interaction*, pp. 294–333. Free Press, New York.

Jefferson, G. (1984). Notes on a systematic deployment of the acknowledgement tokens 'yeah' and 'mm hm'. *Papers in Linguistics*, *17*(2), 197–216.

Jekat, S., Klein, A., Maier, E., Maleck, I., Mast, M., and Quantz, J. (1995). Dialogue acts in verbmobil. Verbmobil–Report–65–95.

Labov, W. and Fanshel, D. (1977). *Therapeutic Discourse*. Academic Press.

Lemon, O., Georgila, K., Henderson, J., and Stuttle, M. (2006). An ISU dialogue system exhibiting reinforcement learning of dialogue policies: Generic slot-filling in the TALK in-car system. In *EACL-06*.

Levesque, H. J., Cohen, P. R., and Nunes, J. H. T. (1990). On acting together. In *AAAI-90*, Boston, MA, pp. 94–99. Morgan Kaufmann.

Levin, E., Pieraccini, R., and Eckert, W. (2000). A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, *8*, 11–23.

Levinson, S. C. (1983). *Conversational Analysis*, chap. 6. Cambridge University Press.

Levow, G.-A. (1998). Characterizing and recognizing spoken corrections in human-computer dialogue. In *COLING-ACL*, pp. 736–742.

Litman, D. J., Swerts, M., and Hirschberg, J. (2000). Predicting automatic speech recognition performance using prosodic cues. In *NAACL 2000*.

Litman, D. J., Walker, M. A., and Kearns, M. (1999). Automatic detection of poor speech recognition at the dialogue level. In *ACL-99*, College Park, MA, pp. 309–316.

Lochbaum, K. E., Grosz, B. J., and Sidner, C. L. (2000). Discourse structure and intention recognition. In Dale, R., Moisl, H., and Somers, H. L. (Eds.), *Handbook of Natural Language Processing*. Marcel Dekker.

Mrkšić, N., O'Séaghdha, D., Wen, T.-H., Thomson, B., and Young, S. (2017). Neural belief tracker: Data-driven dialogue state tracking. In *ACL 2017*.

Nagata, M. and Morimoto, T. (1994). First steps toward statistical modeling of dialogue to predict the speech act type of the next utterance. *Speech Communication*, *15*, 193–203.

Norman, D. A. (1988). *The Design of Everyday Things*. Basic Books.

Oh, A. H. and Rudnicky, A. I. (2000). Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pp. 27–32.

Perrault, C. R. and Allen, J. (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, *6*(3-4), 167–182.

Purver, M. (2004). *The theory and use of clarification requests in dialogue*. Ph.D. thesis, University of London.

Roy, N., Pineau, J., and Thrun, S. (2000). Spoken dialog management for robots. In *ACL-00*, Hong Kong.

Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach* (2nd Ed.). Prentice Hall.

Sacks, H., Schegloff, E. A., and Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, *50*(4), 696–735.

Sag, I. A. and Liberman, M. Y. (1975). The intonational disambiguation of indirect speech acts. In *CLS-75*, pp. 487–498. University of Chicago.

Schegloff, E. A. (1968). Sequencing in conversational openings. *American Anthropologist*, *70*, 1075–1095.

Schegloff, E. A. (1972). Notes on a conversational practice: Formulating place. In Sudnow, D. (Ed.), *Studies in social interaction*, New York. Free Press.

Schegloff, E. A. (1982). Discourse as an interactional achievement: Some uses of 'uh huh' and other things that come between sentences. In Tannen, D. (Ed.), *Analyzing Discourse: Text and Talk*, pp. 71–93. Georgetown University Press, Washington, D.C.

Shriberg, E., Bates, R., Taylor, P., Stolcke, A., Jurafsky, D., Ries, K., Coccaro, N., Martin, R., Meteer, M., and Van Ess-Dykema, C. (1998). Can prosody aid the automatic classification of dialog acts in conversational speech?. *Language and Speech (Special Issue on Prosody and Conversation)*, *41*(3-4), 439–487.

Singh, S. P., Litman, D. J., Kearns, M., and Walker, M. A. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*, *16*, 105–133.

Stalnaker, R. C. (1978). Assertion. In Cole, P. (Ed.), *Pragmatics: Syntax and Semantics Volume 9*, pp. 315–332. Academic Press.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Meteer, M., and Van Ess-Dykema, C. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, *26*(3), 339–371.

Stoyanchev, S. and Johnston, M. (2015). Localized error detection for targeted clarification in a virtual assistant. In *ICASSP-15*, pp. 5241–5245.

Stoyanchev, S., Liu, A., and Hirschberg, J. (2013). Modelling human clarification strategies. In *SIGDIAL 2013*, pp. 137–141.

Stoyanchev, S., Liu, A., and Hirschberg, J. (2014). Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Bradford Books (MIT Press).

Swerts, M., Litman, D. J., and Hirschberg, J. (2000). Corrections in spoken dialogue systems. In *ICSLP-00*, Beijing, China.

Wade, E., Shriberg, E., and Price, P. J. (1992). User behaviors affecting speech recognition. In *ICSLP-92*, pp. 995–998.

Walker, M. A. (2000). An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, *12*, 387–416.

Walker, M. A., Fromer, J. C., and Narayanan, S. S. (1998). Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email. In *COLING/ACL-98*, Montreal, Canada, pp. 1345–1351.

Ward, N. and Tsukahara, W. (2000). Prosodic features which cue back-channel feedback in English and Japanese. *Journal of Pragmatics*, *32*, 1177–1207.

Weinschenk, S. and Barker, D. T. (2000). *Designing Effective Speech Interfaces*. Wiley.

Wen, T.-H., Gasic, M., Kim, D., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. (2015a). Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGDIAL 2015*, pp. 275—284.

Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. (2015b). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP 2015*.

Wilensky, R. (1983). *Planning and Understanding: A Computational Approach to Human Reasoning*. Addison-Wesley.

Williams, J., Raux, A., and Henderson, M. (2016). The dialog state tracking challenge series: A review. *Dialogue & Discourse*, *7*(3), 4–33.

Williams, J. D. and Young, S. J. (2007). Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, *21*(1), 393–422.

Wittgenstein, L. (1953). *Philosophical Investigations. (Translated by Anscombe, G.E.M.)*. Blackwell.

Yankelovich, N., Levow, G.-A., and Marx, M. (1995). Designing SpeechActs: Issues in speech user interfaces. In *Human Factors in Computing Systems: CHI '95 Conference Proceedings*, Denver, CO, pp. 369–376.

Yngve, V. H. (1970). On getting a word in edgewise. In *CLS-70*, pp. 567–577. University of Chicago.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, *24*(2), 150–174.