Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Daniel Jurafsky & James H. Martin. Copyright © 2005, All rights reserved. Draft of May 18, 2005. Do not cite without permission.

19 DIALOGUE AND CONVERSATIONAL AGENTS

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.

The literature of the fantastic abounds in inanimate objects magically endowed with sentience and the gift of speech. From Ovid's statue of Pygmalion to Mary Shelley's Frankenstein, Cao Xue Qin's Divine Luminescent Stone-in-Waiting to Snow White's mirror, there is something deeply touching about creating something and then having a chat with it. Legend has it that after finishing his sculpture of *Moses*, Michelangelo thought it so lifelike that he tapped it on the knee and commanded it to speak. Perhaps this shouldn't be surprising. Language itself has always been the mark of humanity and sentience, and **conversation** or **dialogue** is the most fundamental and specially privileged arena of language. It is certainly the first kind of language we learn as children, and for most of us, it is the kind of language we most commonly indulge in, whether we are ordering curry for lunch or buying postage stamps, participating in business meetings or talking with our families, booking airline flights or complaining about the weather.

CONVERSATION
DIALOGUE

This chapter introduces the fundamental structures and algorithms in **conversational agents**, also known as **spoken dialogue systems**, or **spoken language systems**. These are programs which communicate with users in spoken natural language in order to make travel arrangements, answer questions about weather or sports, route telephone calls, act as a general telephone assistant, or perform even more sophisticated tasks.

Many of these tasks have to do with telephony; speech dialogue systems are particularly appropriate for applications like telephony, where only a limited keyboard and/or screen is available (Cohen and Oviatt, 1995). One important domain for conversational agents is travel planning, which has been a key concern of dialogue systems since the early influential GUS system for planning airline travel (Bobrow et al., 1977). More recently, a number of spoken language systems for booking airline flights were built under the Air Traffic Information System (ATIS) (Hemphill et al., 1990) and Communicator (Walker, 2001) programs. We will use this travel domain as an example throughout the chapter (as we used it to discuss grammar rules for parsing ATIS sentences in Chapter 9). Figure 19.1 shows a sample dialogue from a Communicator system, taken from Xu and Rudnicky (2000).

S1:	Hello. You've reached the [Communicator]. Tell me your full name
U2:	*hi i'd like to fly to seattle tuesday morning*
S3:	Traveling to Seattle on Tuesday, August eleventh in the morning. Your full name?
U4:	*[John Doe]*
S5:	Hello, [John], I'm glad to hear from you again. You can now ask me for help.
S6:	OK. Let's see I have a United flight departing Pittsburgh at ten forty five a.m. arriving into Seattle at one fifty seven p.m. Is that OK?
U7:	*Yes*
S8:	Will you return to Pittsburgh from Seattle?
U9:	*no i will go to Los Angeles*
S10:	Traveling to Los Angeles. on what date
U11:	*Sorry can we go back to talk about the previous leg again*
S12:	For the first leg, you are on a United flight 1446 departing Pittsburgh at ten forty five A.M. tomorrow, arriving in Seattle at one fifty seven P.M. Do you want to take that?
U13:	*actually i want to leave on wednesday*
S14:	a flight on wednesday.
S15:	I have a flight departing Pittsburgh at ten forty five a.m., arrives Seattle at one fifty seven p.m. Is that OK?
U16:	*Yes*

Figure 19.1 The travel domain: a fragment from a successful conversation between a user (U) and the Communicator system (S) of Xu and Rudnicky (2000).

Another promising domain is automatic call routing. A call routing system directs incoming calls in a telephone call center, transferring the call to the appropriate human. Call routing dialogue systems generally begin by asking the user open-ended questions like *How may I help you?* (Gorin et al., 1997), or *How may I direct your call?* (Chu-Carroll and Carpenter, 1999). Figure 19.2 shows sample conversations from the AT&T How May I Help You (HMIHY) system (Gorin et al., 1997).

S:	How may I help you?
U:	*can you tell me how much it is to tokyo?*
S:	You want to know the cost of a call?
U:	*yes that's right*
S:	Please hold on for rate information.
S:	How may I help you?
U:	*what's the area code for chicago?*
S:	You want an area code.
U:	*(silence)*
S:	The area code for Chicago is 312.

Figure 19.2 The call routing domain: two conversations between a user (U) and the AT&T How May I Help You (HMIHY) system (S) (Gorin et al., 1997).

A somewhat different domain, which is very far from the telephony area, is automatic tutoring systems. Figure 19.3 shows part of a dialogue from the ITSPOKE dialogue system (Litman and Silliman, 2004), which uses the Why2-Atlas text-based tutoring system as its tutoring backend (?).

S:	I didn't understand. What force acts on all objects within close proximity to earth?
U:	*uh the force of gravity*
S:	Fine. Besides the Earth's gravitational force, are there any other forces acting on the person, keys, and elevator after the release?
U:	*no*
S:	Yep. Thus, keep in mind that after the release, the only force on the person, keys, and elevator is the force of gravity. If gravity is the only force acting on an object, will it be moving or staying still?
U:	*it will be moving*
S:	Fine. Not only are the person, keys, and elevator moving, they have only gravitational forces on them. When an object is falling and has only gravitational force on it, it is said to be in what?
U:	*freefall*

Figure 19.3 The tutoring domain: part of an conversation between a student user (U) and the ITSPOKE system (S) of Litman and Silliman (2004).

Section 19.1 starts out with a summary of facts about human conversation, including the idea of turns and utterances, speech acts, grounding, dialogue structure, and conversational implicature. Section 19.2 then introduces simple spoken language systems, introducing in detail each of the components of a typical frame-based dialogue system, followed by an overview in Section 19.3 of the VoiceXML language for implementing these systems. We then turn Section 19.4 to the problem of evaluating system performance. Section 19.5 then introduces the more sophisticated information-state model of conversation. Section 19.6 shows how Markov Decision Processes can provided a motivated probabilistic foundation for conversational action. Finally we discuss some advanced topics, including the BDI (belief-desire-intention) paradigm for dialogue understanding, and a brief mention of issues involved in processing human-human dialogue.

19.1 HUMAN CONVERSATION

Conversation is an intricate and complex joint activity. We begin our discussion of conversational agents by offering a sketch of some of what is known about human conversation.

Turns and Turn-Taking

TURN-TAKING

Dialogue is characterized by **turn-taking**; Speaker A says something, then speaker B, then speaker A, and so on. If having a turn (or “taking the floor”) is a resource to be allocated, what is the process by which turns are allocated? How do speakers know when it is the proper time to contribute their turn?

It turns out that conversation and language itself are structured in such a way as to deal efficiently with this resource allocation problem. One source of evidence for this is the timing of the utterances in normal human conversations. While speakers can overlap each other while talking, it turns out that on average the total amount of overlap is remarkably small; perhaps less than 5% (Levinson, 1983). If speakers aren’t overlapping, do they figure out when to talk by waiting for a pause after the other speaker finishes? This is also very rare. The amount of time between turns is quite small, generally less than a few hundred milliseconds even in multi-party discourse. Since it may take more than this few hundred milliseconds for the next speaker to plan the motor routines for producing their utterance, this means that speakers begin motor planning for their next utterance before the previous speaker has finished. For this to be possible, natural conversation must be set up in such a way that (most of the time) people can quickly figure out **who** should talk next, and exactly **when** they should talk. This kind of turn-taking behavior is generally studied in the field of **Conversation Analysis** (**CA**). In a key conversation-analytic paper, Sacks et al. (1974) argued that turn-taking behavior, at least in American English, is governed by a set of turn-taking rules. These rules apply at a **transition-relevance place**, or **TRP**; places where the structure of the language allows speaker shift to occur. Here is a version of the turn-taking rules simplified from Sacks et al. (1974): CONVERSATION ANALYSIS

(19.1) **Turn-taking Rule.** At each TRP of each turn:

a. If during this turn the current speaker has selected A as the next speaker then A must speak next.

b. If the current speaker does not select the next speaker, any other speaker may take the next turn.

c. If no one else takes the next turn, the current speaker may take the next turn.

There are a number of important implications of rule (19.1) for dialogue modeling. First, subrule (19.1a) implies that there are some utterances by which the speaker specifically selects who the next speaker will be. The most obvious of these are questions, in which the speaker selects another speaker to answer the question. Two-part structures like QUESTION-ANSWER are called **adjacency pairs** (Schegloff, 1968); other adjacency pairs include GREETING followed by GREETING, COMPLIMENT followed by DOWNPLAYER, REQUEST followed by GRANT. We will see that these pairs and the dialogue expectations they set up will play an important role in dialogue modeling. ADJACENCY PAIRS

Subrule (19.1a) also has an implication for the interpretation of silence. While silence can occur after any turn, silence in between the two parts of an adjacency pair is **significant silence**. For example Levinson (1983) notes this example from Atkinson and Drew (1979); pause lengths are marked in parentheses (in seconds): SIGNIFICANT SILENCE

(19.2) A: Is there something bothering you or not?
(1.0)
A: Yes or no?
(1.5)
A: Eh?
B: No.

Since A has just asked B a question, the silence is interpreted as a refusal to respond, or perhaps a **dispreferred** response (a response, like saying "no" to a request, which is stigmatized). By contrast, silence in other places, for example a lapse after a speaker finishes a turn, is not generally interpretable in this way. These facts are relevant for user interface design in spoken dialogue systems; users DISPREFERRED are disturbed by the pauses in dialogue systems caused by slow speech recognizers (Yankelovich et al., 1995).

UTTERANCE

Another implication of (19.1) is that transitions between speakers don't occur just anywhere; the **transition-relevance places** where they tend to occur are generally at **utterance** boundaries. Recall from Chapter 9 that spoken utterances differ from written sentences in a number of ways. They tend to be shorter, are more likely to be single clauses or even just single words, the subjects are usually pronouns rather than full lexical noun phrases, and they include filled pauses and repairs. A hearer must take all this (and other cues like prosody) into account to know where to begin talking.

Speech Acts

The previous section showed that conversation consists of a sequence of turns, each of which consists of one or more utterance. A key insight into conversation due to Wittgenstein (1953) but worked out more fully by Austin (1962), is that an utterance in a dialogue is a kind of **action** being performed by the speaker.

PERFORMATIVE

The idea that an utterance is a kind of action is particularly clear in **performative** sentences like the following:

(19.3) I name this ship the *Titanic*.

(19.4) I second that motion.

(19.5) I bet you five dollars it will snow tomorrow.

SPEECH ACTS

When uttered by the proper authority, for example, (19.3) has the effect of changing the state of the world (causing the ship to have the name *Titanic*) just as any action can change the state of the world. Verbs like *name* or *second* which perform this kind of action are called performative verbs, and Austin called these kinds of actions **speech acts**. What makes Austin's work so far-reaching is that speech acts are not confined to this small class of performative verbs. Austin's claim is that the utterance of any sentence in a real speech situation constitutes three kinds of acts:

- **locutionary act:** the utterance of a sentence with a particular meaning.
- **illocutionary act:** the act of asking, answering, promising, etc., in uttering a sentence.
- **perlocutionary act:** the (often intentional) production of certain effects upon the feelings, thoughts, or actions of the addressee in uttering a sentence.

ILLOCUTIONARY FORCE

For example, Austin explains that the utterance of example (19.6) might have the **illocutionary force** of protesting and the perlocutionary effect of stopping the addressee from doing something, or annoying the addressee.

(19.6) You can't do that.

The term **speech act** is generally used to describe illocutionary acts rather than either of the other two types of acts. Searle (1975b), in modifying a taxonomy of Austin's, suggests that all speech acts can be classified into one of five major classes:

- **Assertives:** committing the speaker to something's being the case (*suggesting, putting forward, swearing, boasting, concluding*).
- **Directives:** attempts by the speaker to get the addressee to do something (*asking, ordering, requesting, inviting, advising, begging*).
- **Commissives:** committing the speaker to some future course of action (*promising, planning, vowing, betting, opposing*).
- **Expressives:** expressing the psychological state of the speaker about a state of affairs *thanking, apologizing, welcoming, deploring*.
- **Declarations:** bringing about a different state of the world via the utterance (including many of the performative examples above; *I resign*, *You're fired*.)

Recent research has focused both on formalizing the definition of each of these kinds of acts, and on extending the notion of speech act to deal with conversational phenomena like grounding, the topic of the next section.

Grounding

The previous section suggested that each turn or utterance could be viewed as an action by a speaker. But dialogue is not a series of unrelated independent acts. Instead, dialogue is a collective act performed by the speaker and the hearer. One implication of joint action is that, unlike in monologue, the speaker and hearer must constantly establish **common ground** (Stalnaker, 1978), the set of things that are mutually believed by both speakers. The need to achieve common ground means that the hearer must **ground** the speaker's utterances, making it clear that the hearer has understood the speaker's meaning and intention.

COMMON GROUND

GROUND

As Clark (1996) points out, 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 would-be elevator traveler that she has successfully called the elevator. Clark 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 in performing an action. If the hearer has problems in understanding, she must indicate these problems to the speaker, again so that mutual understanding can eventually be achieved.

How is closure achieved? Clark and Schaefer (1989) introduce the idea that each joint linguistic act or **contribution** has two phases, called **presentation** and **acceptance**. In the first phase, a speaker presents the hearer with an utterance, performing a sort of speech act. In the acceptance phase, the hearer has to ground the utterance, indicating to the speaker whether understanding was achieved.

CONTRIBUTION

What methods can the hearer (call her B) use to ground the speaker A's utterance? Clark and Schaefer (1989) discuss five main types of methods, ordered from weakest to strongest:

1. **Continued attention:** B shows she is continuing to attend and therefore remains satisfied with A's presentation.
2. **Relevant next contribution:** B starts in on the next relevant contribution.
3. **Acknowledgement:** B nods or says a continuer like *uh-huh*, *yeah*, or the like, or an **assessment** like *that's great*.
4. **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.
5. **Display:** B displays verbatim all or part of A's presentation.

REFORMULATING

COLLABORATIVE COMPLETION

Let's look for examples of these in a human-human dialogue example. We'll be returning to this example throughout the chapter; in order to design a more sophisticated machine dialogue agent, it helps to look at how a human agent performs similar tasks. Figure 19.4 shows part of a dialogue between a human travel agent and a human client.

Utterance A_1, in which the agent repeats *in May*, repeated below in boldface, shows the strongest form of grounding, in which the hearer displays their understanding by repeating verbatim part of the speakers words:

C_1:	. . . I need to travel **in May**.
A_1:	And, what day **in May** did you want to travel?

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

C:	He wants to fly from Boston
A:	**Mm hmm**
C:	to Baltimore Washington International

CONTINUER

BACKCHANNEL

The word *mm-hmm* here is a **continuer**, also often called a **backchannel** or an **acknowledgement token**. A continuer is a short utterance which acknowledges the previous utterance, indicating to the speaker that she is understood and hence cueing the other speaker to continue talking (Jefferson, 1984; Schegloff, 1982; Yngve, 1970).

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

In Clark and Schaefer's third method, the speaker starts in on their relevant next contribution. We see a number of examples of this in the sample dialogue above, for example where the speaker asks a question and the hearer answers it. We mentioned these **adjacency pairs** above; other examples include PROPOSAL followed by ACCEPTANCE or REJECTION, APOLOGY followed by ACCEPTANCE/REJECTION, SUMMONS followed by ANSWER, and so on.

In a more subtle but very important kind of grounding act, 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:

And, what day in May did you want to travel?
...
And you're flying into what city?
...
And what time would you like to leave Pittsburgh?

As we will see in Section 19.5, the notions of grounding and contributions can be combined with speech acts to give a more sophisticated model of joint action in conversation; these more sophisticated models are called **dialogue acts**.

Grounding is just as crucial in human-machine conversation as it is in human conversation. The examples below, from Cohen et al. (2004), suggest how unnatural it sounds when a machine doesn't ground properly. The use of *Okay* makes (19.7) a much more natural response than (19.8) to ground a user's rejection:

(19.7) System: Did you want to review some more of your personal profile?
Caller: No.
System: *Okay,* what's next?

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

Indeed, this kind of lack of grounding can cause errors. Stifelman et al. (1993) and Yankelovich et al. (1995) found that humans get confused when a conversational system system doesn't give explicit acknowledgements.

Conversational Structure

We have already seen how conversation is structured by adjacency pairs and contributions. Here we'll briefly discuss one aspect of the **overall organization** of a conversation: conversational openings. The openings of telephone conversations, for example, tend to have a 4-part structure (Clark, 1994; ?):

Stage 1: Enter a conversation, with summons-response adjacency pair

Stage 2: Identification

Stage 3: Establish joint willingness to converse

Stage 4: The first topic is raised, usually by the caller.

These four stages appear in the opening of this short task-oriented conversation from Clark (1994).

Stage	Speaker & Utterance
1	A_1: (rings B's telephone)
1,2	B_1: Benjamin Holloway
2	A_1: this is Professor Dwight's secretary, from Polymania College
2,3	B_1: ooh yes –
4	A_1: uh:m . about the: lexicology *seminar*
4	B_1: *yes*

It is common for the person who answers the phone to speak first (since the caller's ring functions as the first part of the adjacency pair) but for the caller to bring up the first topic, as the caller did above concerning the "lexicology seminar". This fact that the caller usually brings up the first topic causes confusion when the answerer brings up the first topic instead; here's an example of this from the British directory enquiry service from Clark (1994):

Customer: (rings)
Operator: Directory Enquiries, for which town please?
Customer: Could you give me the phone number of um: Mrs. um: Smithson?
Operator: Yes, which town is this at please?
Customer: Huddleston.
Operator: Yes. And the name again?
Customer: Mrs. Smithson.

In the conversation above, the operator brings up the topic (*for which town please?*) in her first sentence, confusing the caller, who ignores this topic and brings up her own. This fact that callers expect to bring up the topic explains why conversational agents for call routing or directory information generally use very open prompts like *How may I help you?* or *How may I direct your call?* rather than a directive prompt like *For which town please?*. Open prompts allow the caller to state their own topic, reducing recognition errors caused by customer confusion.

Conversation has many other kinds of structure, including the intricate nature of conversational closings and the wide use of presequences. We will discuss structure based on **coherence** in Section 19.7.

Conversational Implicature

We have seen that conversation is a kind of joint activity, in which speakers produce turns according to a systematic framework, and that the contributions made by these turns include a presentation phase of performing a kind of action, and an acceptance phase of grounding the previous actions of the interlocutor. So far we have only talked about what might be called the ‘infrastructure’ of conversation. But we have so far said nothing about the actual information that gets communicated from speaker to hearer in dialogue.

While Chapter 14 showed how we can compute meanings from sentences, it turns out that in conversation, the meaning of a contribution is often quite a bit extended from the compositional meaning that might be assigned from the words alone. This is because inference plays a crucial role in conversation. The interpretation of an utterance relies on more than just the literal meaning of the sentences. Consider the client’s response C_2 from the sample conversation in Figure 19.4, repeated here:

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.

Notice that the client does not in fact answer the question. The client merely states that he has a meeting at a certain time. The semantics for this sentence produced by a semantic interpreter will simply mention this meeting. What is it that licenses the agent to infer that the client is mentioning this meeting so as to inform the agent of the travel dates?

Now consider another utterance from the sample conversation, this one by the agent:

A_4: ... There's three non-stops today.

Now this statement would still be true if there were seven non-stops today, since if there are seven of something, there are by definition also three. But what the agent means here is that there are three **and not more than three** non-stops today. How is the client to infer that the agent means **only three** non-stops?

These two cases have something in common; in both cases the speaker seems to expect the hearer to draw certain inferences; in other words, the speaker is communicating more information than seems to be present in the uttered words. These kind of examples were pointed out by Grice (1975, 1978) as part of his theory of

IMPLICATURE

conversational implicature. **Implicature** means a particular class of licensed inferences. Grice proposed that what enables hearers to draw these inferences is that

MAXIMS

conversation is guided by a set of **maxims**, general heuristics which play a guiding role in the interpretation of conversational utterances. He proposed the following four maxims:

QUANTITY

- **Maxim of Quantity:** Be exactly as informative as is required:
  1. Make your contribution as informative as is required (for the current purposes of the exchange).
  2. Do not make your contribution more informative than is required.

QUALITY

- **Maxim of Quality:** Try to make your contribution one that is true:
  1. Do not say what you believe to be false.
  2. Do not say that for which you lack adequate evidence.

RELEVANCE

- **Maxim of Relevance:** Be relevant.

MANNER

- **Maxim of Manner:** Be perspicuous:
  1. Avoid obscurity of expression.
  2. Avoid ambiguity.
  3. Be brief (avoid unnecessary prolixity).
  4. Be orderly.

It is the Maxim of Quantity (specifically Quantity 1) that allows the hearer to know that *three non-stops* did not mean *seven non-stops*. This is because the hearer assumes the speaker is following the maxims, and thus if the speaker meant seven non-stops she would have said seven non-stops ("as informative as is required"). The Maxim of Relevance is what allows the agent to know that the client wants to travel by the 12th. The agent assumes the client is following the maxims, and hence would only have mentioned the meeting if it was relevant at this point in the dialogue. The most natural inference that would make the meeting relevant is the inference that the client meant the agent to understand that his departure time was before the meeting time.

19.2 Basic Dialogue Systems

Now that we've seen a bit about how human dialogue works, let's describe the spoken dialogue systems used in commercial applications today. Figure 19.5 shows a typical architecture for a dialogue system. It has six components. The speech recognition and understanding components extract meaning from the input, while the generation and TTS components map from meaning to speech. The dialog manager controls the whole process, along with a task manager which has knowledge about the task domain (such as air travel). We'll go through each of these components in the next sections. Then we'll explore more sophisticated research systems in following sections.

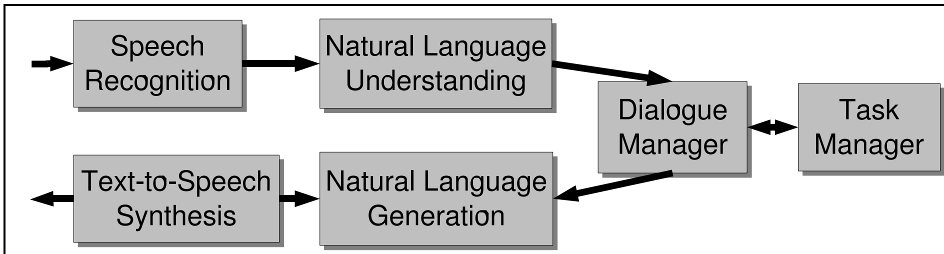

Figure 19.5 Simplified architecture of the components of a conversational agent.

ASR component

The ASR (automatic speech recognition) component takes audio input, generally from the telephone, and returns a transcribed string of words, as discussed in chapters Chapter 6 through Chapter 7. The ASR system may also be optimized in various ways for use in conversational agents. For example while ASR systems used for dictation or transcription generally use a single broadly-trained N-gram language model, ASR systems in conversational agent generally use language models that are specific to a dialogue state. For example, if the system has just asked the user "What city are you departing from?", the ASR language model can be constrained to only consist of city names, or perhaps sentences of the form 'I want to (leave|depart) from [CITYNAME]'. These dialogue-state-specific language models can consist of hand-written finite-state or context-free grammars, or of *N*-gram grammars trained on subcorpora extracted from the answers to particular questions in some training set. When the system wants to constrain the user to respond to the system's last utterance, it can use such a **restrictive grammar**. When the system wants to allow the user more options, it might mix this state-specific language model with a more general language model. As we will see, the choice between these strategies can be tuned based on how much *initiative* the user is allowed.

RESTRICTIVE GRAMMAR

For tasks where the possible things the user is allowed to say are extremely limited, commercial dialogue systems often do not use *N*-gram language models at all. Instead, they use non-probabilistic language models based on finite-state grammars. These grammars are generally hand-written, and specify all possible responses that the system understands. We'll see an example of such a hand-written grammar for a VoiceXML system in Section 19.3

Another way that ASR is influenced by being embedded in a dialogue system has to do with adaptation. Since the identity of the user remains constant across the telephone call, speaker adaptation techniques can be applied to improve recognition as the system hears more and more speech from the user. Thus techniques like MLLR and VTLN (Chapter ASR) can provide useful improvements in ASR rates in a dialogue situation.

NLU component

The NLU (natural language understanding) component of dialogue systems must produce a semantic representation which is appropriate for the dialogue task. Many speech-based dialogue systems, since as far back as the GUS system (Bobrow et al., 1977), are based on the frame-and-slot semantics discussed in Chapter 15. A travel system, for example, which has the goal of helping a user find an appropriate flight, would have a frame with slots for information about the flight; thus a sentence like *Show me morning flights from Boston to San Francisco on Tuesday* might correspond to the following filled-out frame (from Miller et al. (1994)):

```
SHOW:
FLIGHTS:
    ORIGIN:
        CITY:  Boston
        DATE:
            DAY-OF-WEEK:  Tuesday
        TIME:
            PART-OF-DAY:  morning
    DEST:
        CITY:  San Francisco
```

How does the NLU component generate this semantics? In principle any of the methods for semantic analysis discussed in Chapter 15 could be employed. For example, in the SRI GEMINI NLU engine, used in the ATIS and WITAS dialogue systems (?, ?), semantic attachments are added to a unification grammar. A parser produces a sentence meaning, from which the slot-fillers are extracted.

In practice, most dialogue systems rely on simpler domain-specific semantic analyzers, such as the **semantic grammars** also discussed in Chapter 15. In a semantic grammar, the actual node names in the parse tree correspond to the semantic entities which are being expressed, as in the following grammar fragments:

SHOW	→	show me | i want | can i see|...
DEPART_TIME_RANGE	→	(after|around|before) HOUR | morning | afternoon | evening
HOUR	→	one|two|three|four...|twelve (AMPM)
FLIGHTS	→	(a) flight | flights
AMPM	→	am | pm
ORIGIN	→	from CITY
DESTINATION	→	to CITY
CITY	→	Boston | San Francisco | Denver | Washington

These grammars take the form of context-free grammars, and hence can be parsed by any standard CFG parsing algorithm, such as the CKY or Earley algorithms introduced in Chapter 10. In fact, since these domain-specific dialogue system grammars are often simple enough to have no recursion, they can be processed by more efficient finite-state methods. In cases where there is some recursion, efficient augmentations of finite-state algorithms such as recursive transition networks have been applied (Issar and Ward, 1993; Ward and Issar, 1994). The result of the CFG or RTN parse is a hierarchical labeling of the input string with semantic node labels:

```
SHOW     FLIGHTS       ORIGIN  DESTINATION       DEPART_DATE  DEPART_TIME
Show me  flights  from boston  to san francisco  on tuesday   morning
```

Since semantic grammar nodes like ORIGIN correspond to the slots in the frame, the slot-fillers can be read almost directly off the resulting parse above. It remains only to put the fillers into some sort of canonical form (for example dates can be **normalized** into a DD:MM:YY form, times can be put into 24-hour time, etc). NORMALIZED

The semantic grammar approach is very widely used, but has two weaknesses: discreteness (since it is non-probabilistic it has no ambiguity-resolution method) and hand-coding (hand-written grammars are expensive and slow to create).

The discreteness problem can be solved by adding probabilities to the grammar; one such probabilistic semantic grammar system is the TINA system (Seneff,

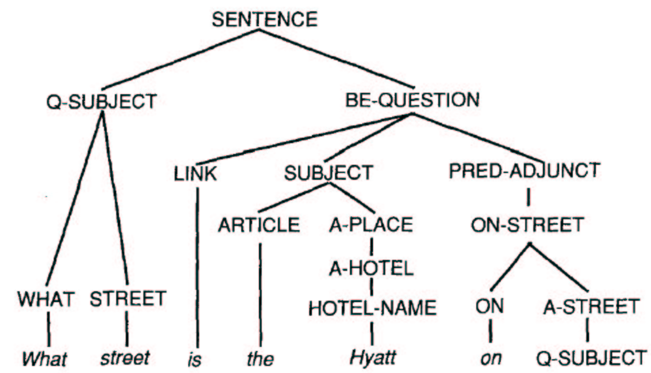

Figure 19.6 A parse of a sentence in the TINA semantic grammar (from (Seneff, 1995)). PLACEHOLDER FIGURE.

1995) shown in Figure 19.6; note the mix of syntactic and semantic node names. The grammar rules in TINA are written by hand, but parse tree node probabilities are trained by a modified version of the SCFG method described in Chapter 12. Instead of conditioning non-terminals just on the parent node, they are also conditioned on the previous non-terminal; this amounts to computing an N-gram grammars of non-terminals conditioned on the parent non-terminal.

$$\begin{aligned} P(\text{ARTICLE}, \text{A-PLACE} | \text{SUBJECT}) \; = \\ P(\text{ARTICLE} | \text{SUBJECT}, < \text{START} >) \times P(\text{A-PLACE} | \text{SUBJECT}, ARTICLE) \end{aligned} \tag{19.9}$$

An alternative to semantic grammars which addresses both the discreteness and hand-coding problems is the semantic HMM model of Pieraccini et al. (1991). The hidden states of this HMM are semantic slot labels, while the observed words are the fillers of the slots. Figure 19.18 shows how a sequence of hidden states, corresponding to slot names, could be decoded from (or could generate) a sequence of observed words. Note that the model includes a hidden state called DUMMY which is used to generate words which do not fill any slots in the frame.

The goal of the HMM model is to compute the labeling of semantic roles $C = c_1, c_2, ..., c_i$ (C for 'cases' or 'concepts') that has the highest probability $P(C|W)$ given some words $W = w_1, w_2, ..., w_n$. As usual, we use Bayes Rule as follows:

$$\begin{aligned} \operatorname*{argmax}_C P(C|W) &= \operatorname*{argmax}_C \frac{P(W|C)P(C)}{P(W} \\ &= \operatorname*{argmax}_C P(W|C)P(C) \end{aligned} \tag{19.10}$$

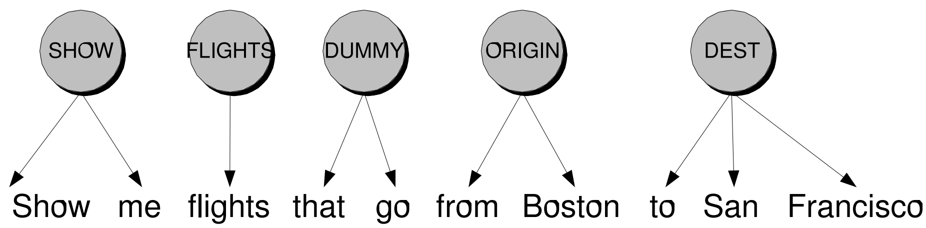

Figure 19.7 The Pieraccini et al. (1991) HMM model of semantics for filling slots in frame-based dialogue systems.

$$= \prod_{i=2}^{N} P(w_i|w_{i-1}...w_1,C)P(w_1|C)\prod_{i=2}^{M} P(c_i|c_{i-1}...c_1) \quad (19.11)$$

The Pieraccini et al. (1991) model makes a simplification that the concepts (the hidden states) are generated by a Markov process (a concept M-gram model), and that the observation probabilities for each state are generated by a state-dependent (concept-dependent) word N-gram word model:

$$P(w_i|w_{i-1},...,w_1,C) = P(w_i|w_{i-1},...,w_{i-N+1},c_i) \quad (19.12)$$

$$P(c_i|c_{i-1},...,c_1) = P(c_i|c_{i-1},...,c_{i-M+1}) \quad (19.13)$$

Based on this simplifying assumption, the final equations used in the HMM model are as follows:

$$\operatorname*{argmax}_{C} P(C|W) = \prod_{i=2}^{N} P(w_i|w_{i-1}...w_{i-N+1},c_i)\prod_{i=2}^{M} P(c_i|c_{i-1}...c_{i-M+1}) \quad (19.14)$$

These probabilities can be trained on a labeled training corpus, in which each sentence is hand-labeled with the concepts/slot-names associated with each string of words. The best sequence of concepts for a sentence, and the alignment of concepts to word sequences, can be computed by the standard Viterbi decoding algorithm.

In summary, the resulting HMM model is a generative model with two components. The $P(C)$ component represents the choice of what meaning to express; it assigns a prior over sequences of semantic slots, computed by a concept N-gram. $P(W|C)$ represents the choice of what words to use to express that meaning; the likelihood of a particular string of words being generated from a given slot. It is computed by a word N-gram conditioned on the semantic slot. This model is very similar to the HMM model for **named entity** detection we saw in Chapter 15.

One problem with the semantic HMM model, as Young (2002) points out, is that it suffers from a data fragmentation problem. Note in Figure 19.18 that each city is labeled as either an *Origin* or a *Destination*. Each city name in the training data can only count as training data for one of the two classes. What we'd like is a way to capture the fact that Boston is the origin and San Francisco the destination, while simultaneously modeling the fact that both are cities. This suggests the need for a hierarchical model.

A third approach to semantic understanding, the Hidden Understanding Model (HUM), attempts solve this problem by combining the advantages of the semantic grammar and semantic HMM approaches (Miller et al., 1994, 1996, 2000). Like the HMM, it doesn't require hand-written grammar rules. But like the semantic grammar, it has the ability to model the hierarchical nature of language structure. The HUM is based on stochastic recursive transition networks (SRTNs), allowing the semantic labels to have hierarchy and recursion. Recall that a recursive transition network is a notational variant of a context-free grammar. Figure 19.8 shows a representation of the HUM structure of the sentence *'Show me United flight 203 from Dallas to Atlanta'*.

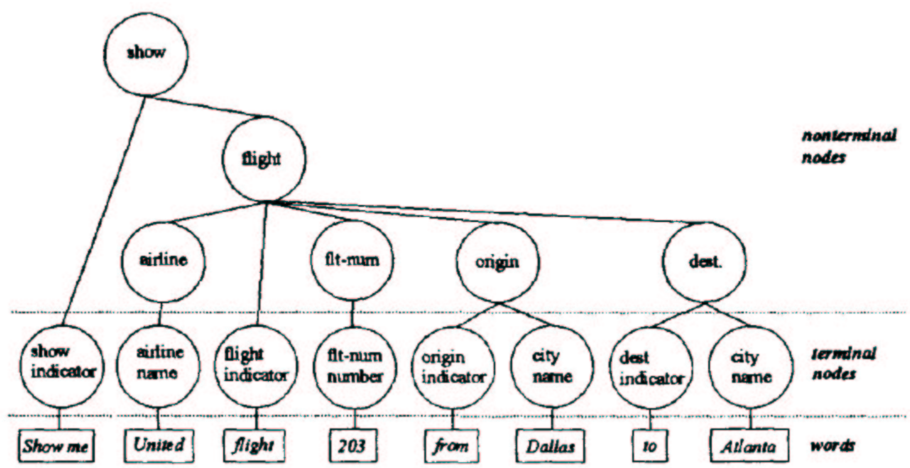

Figure 19.8 Computation of P(C) via the structure of a sentence in the Hidden Understanding Model. (from (Miller et al., 1994)) PLACEHOLDER FIGURE.

The model for $P(W|C)$ in the HUM model is exactly the same as in the HMM model described above: a concept-specific word N-gram model. The model for $P(C)$ is different; instead of using a flat N-gram model of concepts, the HUM uses a modified SCFG model of concept probabilities, following TINA Seneff (1995). Both TINA and HUM assign probabilities based on the state-conditioned N-grams discussed above. The difference between TINA and HUM lies in the role of the hand-written grammar. In TINA, the grammar rules are written by hand, and thus the allowable sequence of non-terminals is prespecified, and includes both syntactic and semantic nodes. In the HUM model the non-terminal sequences are solely semantic, expressing frame slots, and they come from a probabilistic finite-state network (RTN) of concepts in which any concept can follow any other; the only ordering constraint comes from the network probabilities, trained on a labeled training set. Figure 19.9 shows one of the subnetworks for the ATIS *flight* concept; the *flight* node probabilistically generates a sequence of slots (date, origin, airline etc); The arcs on this network represent the (bigram) transition probability of one slot following another, conditioned on the parent node, such as $P(\text{flight ind}|\text{airline, flight})$. The individual nodes like *airline* act as recursive *jump* arcs in the RTN, calling a subnetwork for the airline concept.

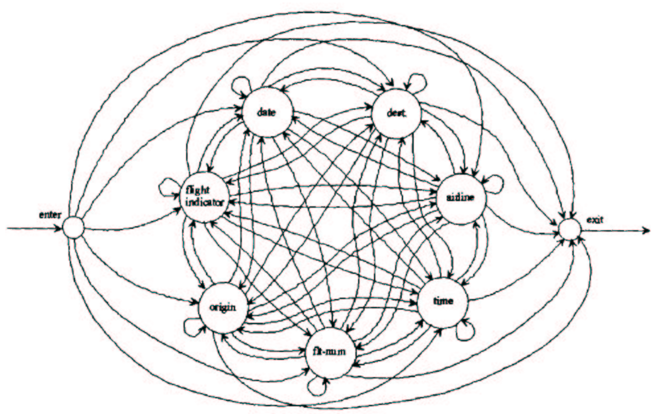

Figure 19.9 The computation of $P(C)$ from the Probabilistic RTN corresponding to the Flight concept, from (Miller et al., 1994). PLACEHOLDER FIGURE.

As with the HMM, HUM decoding (choosing the most likely sequence of concepts for a given sentence) can be done via the Viterbi algorithm. Since the network is a recursive transition network, states must be generated dynamically during the search.

Generation and TTS components

The generation component of a conversational agent chooses the concepts to express to the user, plans out how to express these concepts in words, and assigns any necessary prosody to the words, as described in Chapter 20. The TTS component then takes these words and their prosodic annotations and synthesizes a waveform, as described in Chapter TTS. Both these components may be optimized in various ways for use in conversational agents.

As Chapter 20 describes, the generation task can be separated into two tasks: *what to say*, and *how to say it*. The **content planner** module addresses the first task, decides what content to express to the user, whether to ask a question, present an answer, and so on. The content planning component of dialogue systems is often merged with the dialogue manager.

The **language generation** module addresses the second task, choosing the syntactic structures and words needed to express the meaning. Language generation modules are implemented in one of two ways. In the simplest and most common method, all or most of the words in the sentence to be uttered to the user are prespecified by the dialogue designer. This method is known as template-based generation. While most of the words in the template are fixed, templates can include some variables which are filled in by the generator, as in the following:

What time do you want to leave CITY-ORIG?
Will you return to CITY-ORIG from CITY-DEST?

A second method for language generation relies on the **natural language generation** techniques covered in Chapter 20. Here the dialogue manager builds a representation of the meaning of the utterance to be expressed, and passes this meaning representation to a full generator. Such generators generally have three components, a sentence planner, surface realizer, and prosody assigner. A sketch of this architecture is shown in Figure 19.10.

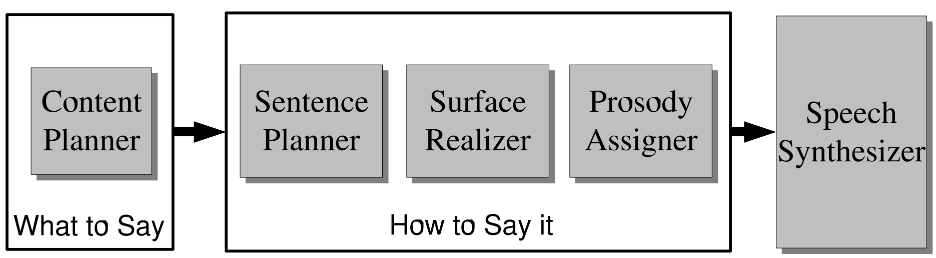

Figure 19.10 Architecture of a generator for a dialogue system, after Walker and Rambow (2002).

Whichever method is used, conversational dialogue places a number of constraints on the sentence generator related to Human Computer Interaction (HCI). Some of these constraints are not that different than other kinds of generation, and reflect the kind of discourse coherence discussed in Chapter 18. For example, as Cohen et al. (2004) show, the use of discourse markers and pronouns makes the dialogue in (19.16) more natural than the dialogue in (19.15):

(19.15) Please say the data.
...

Please say the start time.
...
Please say the duration.
...
Please say the subject.

(19.16) First, tell me the date.
...
Next, I'll need the time it starts.
...
Thanks. <pause> Now, how long is it supposed to last?
...
Last of all, I just need a brief description...

Another important case of discourse coherence occurs when particular prompts may need to be said to the user repeatedly. In these cases, it is standard in dialogue systems to use **tapered prompts**, prompts which get incrementally shorter. The following example from Cohen et al. (2004) shows a series of tapered prompts:

TAPERED PROMPTS

(19.17) System: Now, what's the first company to add to your watch list?
Caller: Cisco
System: What's the next company name? (Or, you can say, "Finished.")
Caller: IBM
System: Tell me the next company name, or say, "Finished."
Caller: Intel
System: Next one?
Caller: America Online.
System: Next?
Caller: ...

Other constraints on generation are more specific to spoken dialogue, and refer to facts about human memory and attentional processes. For example, when humans are prompted to give a particular response, it taxes their memory less if the suggested response is the last thing they hear. Thus as Cohen et al. (2004) point out, the prompt "To hear the list again, say 'Repeat list'" is easier for users than "Say 'Repeat list' to hear the list again."

Similarly, presentation of long lists of query results (e.g., potential flights, or movies) can tax users. Thus most dialogue systems have content planning rules to deal with this. In the Mercury system for travel planning described in (Seneff, 2002), for example, a content planning rule specifies that if there are more than three flights to describe to the user, the system will just list the available airlines and describe explicitly only the earliest flight.

Dialogue Manager

The final component of a dialogue system is the dialogue manager, which controls the architecture and structure of the dialogue. The dialogue manager takes input from the ASR/NLU components, maintains some sort of state, interfaces with the task manager, and passes output to the NLG/TTS modules.

We saw a very simple dialogue manager in Chapter 2's ELIZA, whose architecture was a simple read-substitute-print loop. The system read in a sentence, applied a series of text transformations to the sentence, and then printed it out. No state was kept; the transformation rules were only aware of the current input sentence. What makes a modern dialogue manager very different than ELIZA is both amount of state that the manager keeps about the conversation, and the ability of the manager to model structures of dialogue above the level of a single response.

Four kinds of dialogue management architectures are most common. The simplest and most commercially developed architectures, finite-state and frame-based, are discussed in this section. Later sections discuss the more powerful information-state dialogue managers, including a probabilistic version of information-state managers based on Markov Decision Processes, and finally the more classic plan-based architectures.

The simplest dialogue manager architecture is a finite-state manager. For example, imagine a trivial airline travel system whose job was to ask the user for a departure city, a destination city, a time, and whether the trip was round-trip or not. Figure 19.11 shows a sample dialogue manager for such a system. The states of the FSA correspond to questions that the dialogue manager asks the user, and the arcs correspond to actions to take depending on what the user responds. This system completely controls the conversation with the user. It asks the user a series of questions, ignoring (or misinterpreting) anything the user says that is not a direct answer to the system's question, and then going on to the next question.

SYSTEM INITIATIVE
SINGLE INITIATIVE
INITIATIVE

Systems that control the conversation in this way are called **system initiative** or **single initiative** systems. We say that the speaker that is in control of the conversation has the **initiative**. In normal human-human dialogue, initiative shifts back and forth between the participants (Walker and Whittaker, 1990). The limited single-initiative finite-state dialogue manager architectures may be sufficient for very simple tasks (perhaps for entering a credit card number, or a name and password, on the phone). Furthermore, they have the advantage that the system always knows what question the user is answering. This means the system can prepare the speech recognition engine with a specific language model tuned to answers for this question. Knowing what the user is going to be talking about also makes the task of the natural language understanding engine easier. Pure system-initiative finite-state dialogue manager architectures are probably too restricted, however, even for

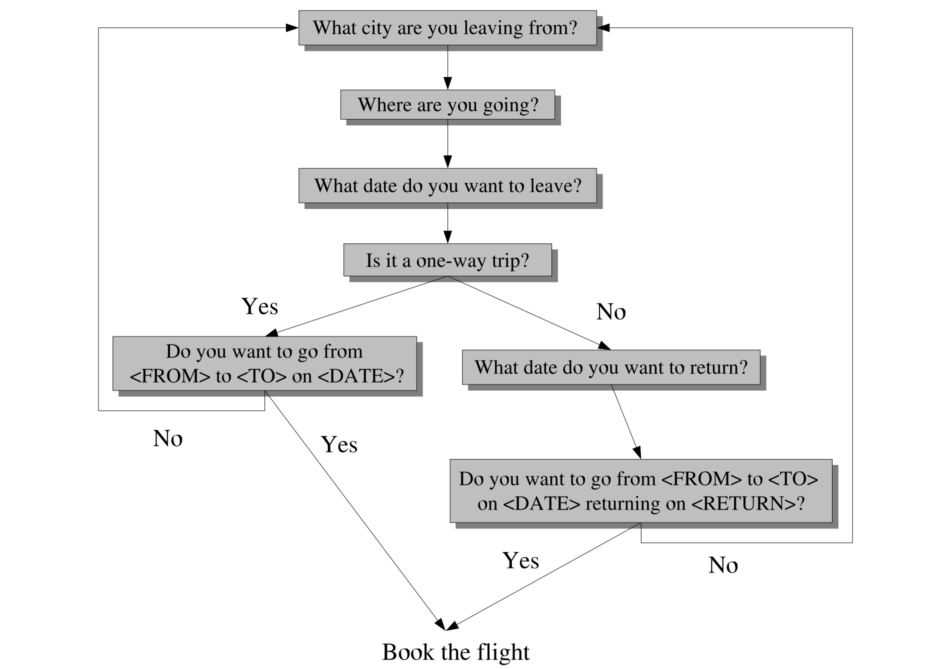

Figure 19.11 A simple finite-state automaton architecture for a dialogue manager.

the relatively uncomplicated task of a spoken dialogue travel agent system.

Single initiative systems can also be controlled by the user, in which case they are called **user initiative** systems. Pure user initiative systems are generally used for stateless database querying systems, where the user asks single questions of the system, which the system converts into SQL database queries, and returns the results from some database.

USER INITIATIVE

The problem is that neither of these kinds of single-initiative systems is practical for the majority of problems. Pure system-initiative systems require that the user answer exactly the question that the system asked. But this can make a dialogue awkward and annoying. Users often need to be able to say something that is not exactly the answer to a single question from the system. For example, in a travel planning situation, users often want to express their travel goals with complex sentences that may answer more than one question at a time, as in Communicator example (19.18) repeated from Figure 19.1, or ATIS example (19.19).

(19.18) Hi I'd like to fly to Seattle Tuesday morning

(19.19) I want a flight from Milwaukee to Orlando one way leaving after five p.m. on Wednesday.

A finite state dialogue system, as typically implemented, can't handle these kinds of utterances since it requires that the user answer each question as it is asked. Of course it is theoretically possible to create a finite state architecture which has a separate state for each possible subset of questions that the user's statement could be answering, but this would require a vast explosion in the number of states, making this a difficult architecture to conceptualize.

UNIVERSAL

Most finite-state systems do allow the user to do things other than answer exactly the question which the system asked. The systems allow **universal** commands. Universals are commands that can be said anywhere in the dialog. They are implemented by essentially allowing every state to recognize the universal commands in addition to the answer to the question that the system just asked. Common universals include **help**, which gives the user a (possibly state-specific) help message, **start over** (or **main menu**), which returns the user to some specified main start state, and some sort of command to correct the system's understanding of the users last statement. For example, in the travel system of San-Segundo et al. (2001), when the system misrecognizes a user's utterance, the user can say **correct** and the system will erase the misrecognition and go back.

MIXED INITIATIVE

But adding universals to a system-initiative architecture is still insufficient. Therefore, most systems avoid the pure system-initiative finite-state approach and use an architecture that allows **mixed initiative**, in which conversational initiative can shift between the system and user at various points in the dialogue.

FRAME-BASED
FORM-BASED

One common mixed initiative dialogue architecture relies on the structure of the frame itself to guide the dialogue. These **frame-based** or **form-based** dialogue managers asks the user questions to fill slots in the frame, but allow the user to guide the dialogue by giving information that fills other slots in the frame. Each slot may be associated with a question to ask the user, of the following type:

Slot	**Question**
ORIGIN CITY	"From what city are you leaving?"
DESTINATION CITY	"Where are you going?"
DEPARTURE TIME	"When would you like to leave?"
ARRIVAL TIME	"When do you want to arrive?"

A frame-based dialogue manager thus needs to ask questions of the user, filling any slot that the user specifies, until it has enough information to perform a data base query, and then return the result to the user. If the user happens to answer two or three questions at a time, the system has to fill in these slots and then remember not to ask the user the associated questions for the slots. Not every slot need have an associated question, since the dialogue designer may not want the user deluged with questions. Nonetheless, the system must be able to fill these slots if the user happens to specify them. This kind of form-filling dialogue manager thus does away with the strict constraints that the finite-state manager imposes on the order that the user can specify information.

While some domains may be representable with a single frame, others, like the travel domain, seem to require the ability to deal with multiple frames. In order to handle possible user questions, we might need frames with general route information (for questions like *Which airlines fly from Boston to San Francisco?*), information about airfare practices (for questions like *Do I have to stay a specific number of days to get a decent airfare?*) or about car or hotel reservations. Since users may switch from frame to frame, the system must be able to disambiguate which slot of which frame a given input is supposed to fill, and then switch dialogue control to that frame.

Because of this need to dynamically switch control, frame-based systems are often implemented as **production rule** systems. Different types of inputs cause different productions to fire, each of which can flexibly fill in different frames. The production rules can then switch control based on factors such as the user's input and some simple dialogue history like the last question that the system asked. The Mercury flight reservation system (Seneff and Polifroni, 2000; Seneff, 2002) uses a large 'dialogue control table' to store 200-350 rules, covering request for help, rules to determine if the user is referring to a flight in a list ("I'll take that nine a.m. flight"), and rules to decide which flights to describe to the user first.

Now that we've seen the frame-based architecture, let's return to our discussion of conversational initiative. It's possible in the same agent to allow system-initiative, user-initiative, and mixed-initiative interactions. We said earlier that initiative refers to who has control of the conversation at any point. The phrase **mixed initiative** is generally used in two ways. It can mean that the system or the user could arbitrarily take or give up the initiative in various ways (Walker and Whittaker, 1990; Chu-Carroll and Brown, 1997). This kind of mixed initiative is difficult to achieve in current dialogue systems. In form-based dialogue system, the term mixed initiative is used for a more limited kind of shift, operationalized based on a combination of prompt type (open versus directive) and the type of grammar used in the ASR. An **open prompt** is one in which the system gives the user very few constraints, allowing the user to respond however they please, as in: OPEN PROMPT

How may I help you?

A **directive prompt** is one which explicitly instructs the user how to respond: DIRECTIVE PROMPT

Say *yes* if you accept the call; otherwise, say *no*.

In Section 19.2 we defined a **restrictive** grammar as a language model which strongly constrains the ASR system, only recognizing proper responses to a given prompt.

	Prompt Type	
Grammar	**Open**	**Directive**
Restrictive	*Doesn't make sense*	System Initiative
Non-Restrictive	User Initiative	Mixed Initiative

Figure 19.12 Operational definition of initiative, following Singh et al. (2002).

In Figure 19.12 we then give the definition of initiative used in form-based dialogue systems, following Singh et al. (2002) and others. Here a system initiative interaction uses a directive prompt and a restrictive grammar; the user is told how to respond, and the ASR system is constrained to only recognize the responses that are prompted for. In user initiative, the user is given an open prompt, and the grammar must recognize any kind of response, since the user could say anything. Finally, in a mixed initiative interaction, the system gives the user a directive prompt with particular suggestions for response, but the non-restrictive grammar allows the user to respond outside the scope of the prompt.

Defining initiative as a property of the prompt and grammar type in this way allows systems to dynamically change their initiative type for different users and interactions. Novice users, or users with high speech recognition error, might be better served by more system initiative. Expert users, or those who happen to speak more recognizably, might do well with mixed or user initiative interactions. We will see in Section 19.6 how machine learning techniques can be used to choose initiative.

19.3 VoiceXML

VOICEXML
VXML

VoiceXML is the Voice Extensible Markup Language, an XML-based dialogue design language released by the W3C. The goal of VoiceXML (or **vxml**) is to create simple audio dialogues of the type we have been describing, making use of ASR and TTS, and dealing with very simple mixed-initiative in a frame-based architecture. While VoiceXML is more common in the commercial rather than academic setting, it offers a convenient summary of the dialogue system design issues we have discussed, and will continue to discuss.

A VoiceXML document contains a set of dialogs, each of which can be a *form* or a *menu*. We will limit ourselves to introducing forms; see (?) for more information on VoiceXML in general. The VoiceXML document in Figure 19.13 defines a form with a single field named 'transporttype'. The field has an attached prompt, *Please choose airline, hotel, or rental car*, which can be passed to the TTS system. It also has a grammar (language model) which is passed to the speech

```
<form>
  <field name="transporttype">
     <prompt>
        Please choose airline, hotel, or rental car.
     </prompt>
     <grammar type="application/x=nuance-gsl">
       [airline hotel "rental car"]
     </grammar>
  </field>
  <block>
    <prompt>
     You have chosen <value expr="transporttype">.
    </prompt>
  </block>
</form>
```

Figure 19.13 A minimal VoiceXML script for a form with a single field. User is prompted, and the response is then repeated back.

recognition engine to specify which words the recognizer is allowed to recognize. In the example in Figure 19.13, the grammar consists of a disjunction of the three words *airline*, *hotel*, and *rental car*.

A `<form>` generally consists of a sequence of `<field>`s, together with a few other commands. Each field has a name (the name of the field in Figure 19.13 is `transporttype`) which is also the name of the variable where the user's response will be stored. The prompt associated with the field is specified via the `<prompt>` command. The grammar associated with the field is specified via the `<grammar>` command. VoiceXML supports various ways of specifying a grammar, including XML Speech Grammar, ABNF, and commercial standards, like Nuance GSL. We will be using the Nuance GSL format in the following examples.

The VoiceXML interpreter walks through a form in document order, repeatedly selecting each item in the form. If there are multiple fields, the interpreter will visit each one in order. The interpretation order can be changed in various ways, as we will see later. The example in Figure 19.14 shows a form with three fields, for specifying the origin, destination, and flight date of an airline flight.

The prologue of the example shows two global defaults for error handling. If the user doesn't answer after a prompt (i.e., silence exceeds a timeout threshold), the VoiceXML interpreter will play the `<noinput>` prompt. If the user says something, but it doesn't match the grammar for that field, the VoiceXML interpreter will play the `<nomatch>` prompt. After any failure of this type, it is normal to re-ask the user the question that failed to get a response. Since these routines can be called from any field, and hence the exact prompt will be different every time, VoiceXML provides a `<reprompt\>` command, which will repeat the prompt for whatever field caused the error.

```
<noinput>
I'm sorry, I didn't hear you.  <reprompt/>
</noinput>

<nomatch>
I'm sorry, I didn't understand that.  <reprompt/>
</nomatch>

<form>
  <block>    Welcome to the air travel consultant.  </block>
  <field name="origin">
      <prompt>     Which city do you want to leave from?  </prompt>
      <grammar type="application/x=nuance-gsl">
        [(san francisco) denver (new york) barcelona]
      </grammar>
      <filled>
         <prompt>  OK, from <value expr="origin">  </prompt>
      </filled>
  </field>
  <field name="destination">
      <prompt>  And which city do you want to go to?   </prompt>
      <grammar type="application/x=nuance-gsl">
        [(san francisco) denver (new york) barcelona]
      </grammar>
      <filled>
         <prompt>   OK, to <value expr="destination">   </prompt>
      </filled>
  </field>
  <field name="departdate" type="date">
      <prompt>  And what date do you want to leave?  </prompt>
      <filled>
         <prompt>   OK, on <value expr="departdate">   </prompt>
      </filled>
  </field>
  <block>
      <prompt> OK, I have you are departing from  <value expr="origin">
              to <value expr="destination"> on <value expr="departdate">
      </prompt>
      send the info to book a flight...
  </block>
</form>
```

Figure 19.14 A VoiceXML script for a form with 3 fields, which confirms each field and handles the `noinput` and `nomatch` situations.

The three fields of this form show another feature of VoiceXML, the `<filled>` tag. The `<filled>` tag for a field is executed by the interpreter as soon as the field has been filled by the user. Here, this feature is used to give the user a confirmation of their input.

The last field, `departdate`, shows another feature of VoiceXML, the `type` attribute. VoiceXML 2.0 specifies seven built-in grammar types, `boolean`, `currency`, `date`, `digits`, `number`, `phone`, and `time`. Since the type of this field is `date`, a data-specific language model (grammar) will be automatically passed to the speech recognizer, so we don't need to specify the grammar here explicitly.

```
<noinput>    I'm sorry, I didn't hear you.  <reprompt/>  </noinput>

<nomatch> I'm sorry, I didn't understand that.  <reprompt/> </nomatch>

<form>
   <grammar type="application/x=nuance-gsl">
    <![ CDATA[
    Flight (  ?[
             (i [wanna (want to)] [fly go])
             (i'd like to [fly go])
             ([(i wanna)(i'd like a)] flight)
          ]
          [
            ( [from leaving departing] City:x) {<origin $x>}
            ( [(?going to)(arriving in)] City:x) {<destination $x>}
            ( [from leaving departing] City:x
              [(?going to)(arriving in)] City:y) {<origin $x> <destination $y>}
          ]
          ?please
     )
     City [ [(san francisco) (s f o)] {return( "san francisco, california")}
            [(denver) (d e n)] {return( "denver, colorado")}
            [(seattle) (s t x)] {return( "seattle, washington")}
     ]
     ]]> </grammar>

  <initial name="init">
     <prompt> Welcome to the air travel consultant. What are your travel plans?  </prompt>
  </initial>

  <field name="origin">
     <prompt> Which city do you want to leave from?  </prompt>
     <filled>
        <prompt> OK, from <value expr="origin"> </prompt>
     </filled>
  </field>
  <field name="destination">
     <prompt> And which city do you want to go to?  </prompt>
     <filled>
        <prompt> OK, to <value expr="destination"> </prompt>
     </filled>
  </field>
  <block>
     <prompt> OK, I have you are departing from  <value expr="origin">
              to  <value expr="destination">.  </prompt>
    send the info to book a flight...
  </block>
</form>
```

Figure 19.15 A mixed initiative VoiceXML dialog. The grammar allows sentences which specify the origin or destination cities or both. User can respond to the initial prompt by specifying origin city, destination city, or both.

Figure 19.15 gives a final example which shows mixed initiative. In a mixed initiative dialogue, users can choose not to answer the question that was asked by the system. For example, they might answer a different question, or use a long sentence to fill in multiple slots at once. This means that the VoiceXML interpreter can no longer just evaluate each field of the form in order; it needs to skip fields whose values are set. This is done by a *guard condition*, a test that keeps a field from being visited. The default guard condition for a field tests to see if the field's form item variable has a value, and if so the field is not interpreted.

Figure 19.15 also shows a much more complex use of a grammar. This grammar is a CFG grammar with two rewrite rules, named `Flight` and `City`. The Nuance GSL grammar formalism uses parentheses () to mean concatenation and square brackets [] to mean disjunction. Thus a rule like (19.20) means that `Wantsentence` can be expanded as `i want to fly` or `i want to go`, and `Airports` can be expanded as `san francisco` or `denver`.

(19.20)
```
Wantsentence (i want to [fly go])
Airports [(san francisco) denver]
```

Grammar rules can refer to other grammar rules recursively, and so in the grammar in Figure 19.15 we see the grammar for `Flight` referring to the rule for `City`.

VoiceXML grammars take the form of CFG grammars with optional semantic attachments. The semantic attachments are generally either a text string (such as `"denver, colorado"`) or a slot and a filler. We can see an example of the former in the semantic attachments for the `City` rule (the `return` statements at the end of each line), which pass up the city and state name. The semantic attachments for the `Flight` rule shows the latter case, where the slot (`<origin>` or `<destination>` or both) is filled with the value passed up in the variable *x* from the `City` rule.

Because Figure 19.15 is a mixed initiative grammar, the grammar has to be applicable to any of the fields. This is done by making the expansion for `Flight` a disjunction; note that it allows the user to specify only the origin city, only the destination city, or both.

19.4 DIALOGUE SYSTEM EVALUATION

An optimal dialogue system is one which allows a user to accomplish their goals (maximizing task success) with the least problems (minimizing costs). A number of metrics for each of these two criteria have been proposed.

Task Completion Success: Task success can be measured by evaluating the correctness of the total solution. For a frame-based architecture, this might be the percentage of slots that were filled with the correct values, or the percentage of subtasks that were completed (Polifroni et al., 1992). Since different dialogue sys-

METHODOLOGY BOX: DESIGNING DIALOGUE SYSTEMS

How does a dialogue system developer choose dialogue strategies, architectures, prompts, error messages, and so on? The three design principles of Gould and Lewis (1985) can be summarized as: **User-Centered Design:** Study the user and task, Build simulations and prototypes, and Iteratively test them on the user and fix the problems.

1. Early Focus on Users and Task: Understand the potential users and the nature of the task, via interviews with users and investigation of similar systems, and study of related human-human dialogues.

2. Build Prototypes: In Wizard-of-Oz (WOZ) or PNAMBIC (Pay No Attention to the Man BehInd the Curtain) systems, the users interact with what they think is a software system, but is in fact a human operator ("wizard") behind some disguising interface software (e.g. Gould et al., 1983; Good et al., 1984; Fraser and Gilbert, 1991) . The name comes from the children's book *The Wizard of Oz* (Baum, 1900), in which the Wizard turned out to be just a simulation controlled by a man behind a curtain. A WOZ system can be used to test out an architecture before implementation; only the interface software and databases need to be in place. The wizard's linguistic output can be be disguised by a text-to-speech system, or via text-only interactions. It is difficult for the wizard to exactly simulate the errors, limitations, or time constraints of a real system; results of WOZ studies are thus somewhat idealized, but still can provide a useful first idea of the domain issues.

3. Iterative Design: An iterative design cycle with embedded user testing is essential in system design (Nielsen, 1992; Cole et al., 1994, 1997; Yankelovich et al., 1995; Landauer, 1995). For example Stifelman et al. (1993) and Yankelovich et al. (1995) found that users of speech systems consistently tried to interrupt the system (**barge-in**), suggesting a redesign of the system to recognize overlapped speech. The iterative method is also very important for designing prompts which cause the user to respond in understandable or normative ways: Kamm (1994) and Cole et al. (1993) found that **directive prompts** ("Say *yes* if you accept the call, otherwise, say *no*") or the use of constrained forms (Oviatt et al., 1993) produced better results than open prompts like "Will you accept the call?". Simulations can also be used at this stage; user simulations that interact with a dialogue system can help test the interface for brittleness or errors (Chung, 2004).

See Cohen et al. (2004) for more details on conversational interface design.

tems may be applied to different tasks, it is hard to compare them on this metric, so Walker et al. (1997) suggested using the Kappa coefficient, κ, to compute a completion score which is normalized for chance agreement and better enables cross-system comparison.

Finally, Walker et al. (2001) notes that sometimes the user's *perception* of whether they completed the task is a better predictor of user satisfaction than the above measures. In more recent studies on evaluation of dialogue systems, Walker et al. (2002) gives users an on-line survey after completing a dialogue, which ask for a yes-no answer as to whether the task was completed.

Efficiency Cost: Efficiency costs are measures of the system's efficiency at helping users. This can be measured via the total elapsed time for the dialogue in seconds, the number of total turns or of system turns, or the total number of queries (Polifroni et al., 1992). Other metrics include the number of system non-responses, and the "turn correction ratio": the number of system or user turns that were used solely to correct errors, divided by the total number of turns (Danieli and Gerbino, 1995; Hirschman and Pao, 1993).

Quality Cost: Quality cost measures other aspects of the interaction that affect users' perception of the system. One such measure is the number of times the ASR system failed to return any sentence, or the number of ASR rejection prompts ('I'm sorry, I didn't understand that'). Similar metrics include the number of times the user had to **barge-in** (interrupt the system), or the number of time-out prompts played when the user didn't respond quickly enough. Other quality metrics focus on how well the system understood and responded to the user. This can include the inappropriateness (verbose or ambiguous) of the system's questions, answers, and error messages (Zue et al., 1989), or the correctness of each question, answer, or error message (Zue et al., 1989; Polifroni et al., 1992). A very important quality cost is **concept accuracy** or **concept error rate**, which measures the percentage of semantic concepts that the NLU component returns correctly. For frame-based architectures this can be measured by counting the percentage of slots that are filled with the correct meaning. For example if the sentence 'I want to arrive in Austin at 5:00' is misrecognized to have the semantics "DEST-CITY: Boston, Time: 5:00" the concept accuracy would be 50% (one of two slots are wrong) (?).

BARGE-IN

CONCEPT ACCURACY

How should these success and cost metrics be combined and weighted? The PARADISE algorithm (Walker et al., 1997) (PARAdigm for DIalogue System Evaluation) applies multiple regression to this problem. The algorithm first uses questionnaires to assign each dialogue a user satisfaction rating. A set of cost and success factors like those above is then treated as a set of independent factors; multiple regression is used to train a weight (coefficient) for each factor, measuring its importance in accounting for user satisfaction. Figure 19.16 shows the particular

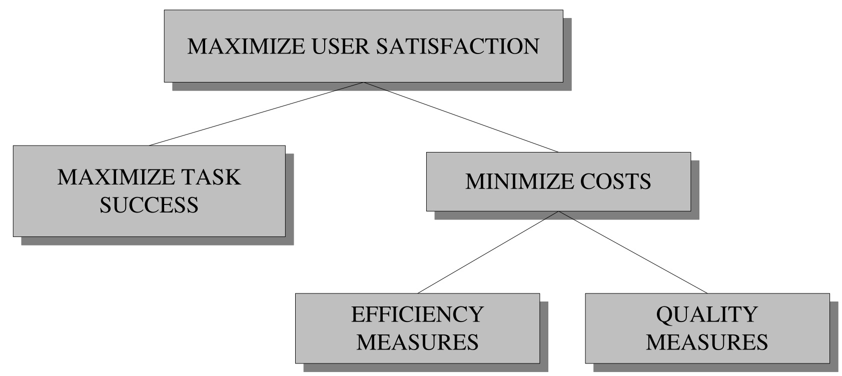

Figure 19.16 PARADISE's structure of objectives for spoken dialogue performance. After Walker et al. (2001).

TTS Performance	Was the system easy to understand ?
ASR Performance	Did the system understand what you said?
Task Ease	Was it easy to find the message/flight/train you wanted?
Interaction Pace	Was the pace of interaction with the system appropriate?
User Expertise	Did you know what you could say at each point?
System Response	How often was the system sluggish and slow to reply to you?
Expected Behavior	Did the system work the way you expected it to?
Future Use	Do you think you'd use the system in the future?

Figure 19.17 User satisfaction survey, adapted from Walker et al. (2001).

model of performance that the PARADISE experiments have assumed. Each box is related to a set of factors that we summarized on the previous page. The resulting metric can be used to compare quite different dialogue strategies.

The user satisfaction rating is computed by having users complete a survey with questions such as those in Figure 19.17, probing their perception of different aspects of the system's performance (Shriberg et al., 1992; Polifroni et al., 1992; Stifelman et al., 1993; Yankelovich et al., 1995). Surveys in PARADISE studies are multiple choice, with the responses mapped into the range of 1 to 5. The scores for each question are then averaged to get a total user satisfaction rating.

Walker et al. (2001, 2002) applied the PARADISE algorithm to three different dialogue systems and found three factors that were often the best predictors of user satisfaction: (1) the average concept accuracy for the whole dialogue, (2) the user's (binary) opinion about whether they completed the task successfully, and (3) the total elapsed time.

19.5 INFORMATION-STATE & DIALOGUE ACTS

The basic frame-based dialogue systems we have introduced so far are only capable of limited domain-specific conversations. This is because the semantic interpretation and generation processes in frame-based dialogue systems are based only on what is needed to fill slots. In order to be be usable for more than just form-filling applications, a conversational agent needs to be able to do things like decide when the user has asked a question, made a proposal, or rejected a suggestion, and needs to be able to ground a users utterance, ask clarification questions, and suggest plans. This suggests that a conversational agent needs sophisticated models of interpretation and generation in terms of speech acts and grounding, and a more sophisticated representation of the dialogue context than just a list of slots.

In this section we sketch a more advanced architecture for dialogue management which allows for these more sophisticated components. This model is generally called the **information-state** architecture (Traum and Larsson, 2003), although we will use the term loosely to include architectures such as Allen et al. (2001). A probabilistic architecture which can be seen as an extension of the information-state approach, the **Markov decision process** model, will be described in the next section. The term **information-state architecture** is really a cover term for a number of quite different efforts toward more sophisticated agents; we'll assume here a structure consisting of 5 components:

INFORMATION-STATE

- the information state (the 'discourse context' or 'mental model')
- a dialogue act interpreter (or "interpretation engine")
- a dialogue act generator (or "generation engine")
- a set of update rules, which update the information state as dialogue acts are interpreted, and which include rules to generate dialogue acts.
- a control structure to select which update rules to apply

The term **information state** is intended to be very abstract, and might include things like the discourse context and the common ground of the two speakers, the beliefs or intentions of the speakers, user models, and so on. Crucially, information state is intended to be a more complex notion than the static states in a finite-state dialogue manager; the current state includes the values of many variables, the discourse context, and other elements that are not easily modeled by a state-number in a finite network.

Dialogue acts are an extension of speech acts which integrate ideas from grounding theory, and will be defined more fully fully in the next subsection. The interpretation engine takes speech as input and figures out sentential semantics and an appropriate dialogue act. The dialogue act generator takes dialogue acts and sentential semantics as input and produces text/speech as output.

Finally, the update rules modify the information state with the information from the dialogue acts. These update rules are a generalization of the production rules used in frame-based dialogue systems described above (Seneff and Polifroni, 2000, *inter alia*). A subset of update rules, called **selection rules**, are used to generate dialogue acts. For example, an update rule might say that when the interpretation engine recognizes an assertion, that the information state should be updated with the information in the assertion, and an obligation to perform a grounding act needs to be added to the information state. When a question is recognized, an update rule might specify the need to answer the question. One way to think about the information-state approach is to view the combination of the update rules and control structure as a kind of "Behavioral Agent" of the kind proposed in the TRIPS system (Allen et al., 2001).

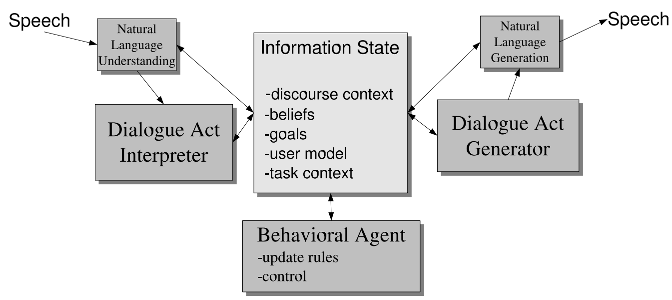

Figure 19.18 A version of the information-state approach to dialogue architecture.

While the intuition of the information-state model is quite simple, the details can be quite complex. The information state might involve rich discourse models such as Discourse Representation Theory or sophisticated models of the user's belief, desire, and intention (which we will return to in Section 19.7). Instead of describing a particular implementation here, we will focus on the dialogue act interpretation and generation engines. The next subsections will present a definition of dialogue acts, a model for detecting them, and a model for generating them. The following section will then show how to use Markov decision processes to implement a probabilistic version of the information-state architecture.

Dialogue Acts

As we implied above, the speech acts as originally defined by Austin don't model key features of conversation such as grounding, contributions, adjacency pairs and so on. In order to capture these conversational phenomena, we use an extension of speech acts called **dialogue acts** (Bunt, 1994) (or **dialogue moves** or **conversa-** DIALOGUE ACT

MOVES

tional moves (Power, 1979; Carletta et al., 1997b). A dialogue act extends speech acts with internal structure related specifically to these other conversational functions (Allen and Core, 1997; Bunt, 2000).

A wide variety of dialogue act tagsets have been proposed. Figure 19.19 shows a very domain-specific tagset for the Verbmobil two-party scheduling domain, in which speakers were asked to plan a meeting at some future date. Notice that it has many very domain-specific tags, such as SUGGEST, used for when someone proposes a particular date to meet, and ACCEPT and REJECT, used to accept or reject a proposal for a date. Thus it has elements both from the presentation and acceptance phases of the Clark contributions discussed on page **??**.

Tag	**Example**
THANK	*Thanks*
GREET	*Hello Dan*
INTRODUCE	*It's me again*
BYE	*Allright 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 19.19 The 18 high-level dialogue acts used in Verbmobil-1, abstracted over a total of 43 more specific dialogue acts. Examples are from Jekat et al. (1995).

A more domain-independent dialogue act tagset is the DAMSL (Dialogue Act Markup in Several Layers) architecture, which draws inspiration from the work on grounding, speech acts, and conversational analysis discussed at the beginning of the chapter. (Allen and Core, 1997; Walker et al., 1996; Carletta et al., 1997a; Core et al., 1999).

For example, drawing on the idea of contributions (Clark and Schaefer, 1989) and the work of Allwood et al. (1992), Allwood (1995), the DAMSL tag set distinguishes between the **forward looking** and **backward looking** function of an utterance. The forward looking function of an utterance extends the Searle/Austin speech act:

Forward Looking Function	
STATEMENT	a claim made by the speaker
INFO-REQUEST	a question by the speaker
CHECK	a question for confirming information
INFLUENCE-ON-ADDRESSEE	(=Searle's directives)
OPEN-OPTION	a weak suggestion or listing of options
ACTION-DIRECTIVE	an actual command
INFLUENCE-ON-SPEAKER	(=Austin's commissives)
OFFER	speaker offers to do something, (subject to confirmation)
COMMIT	speaker is committed to doing something
CONVENTIONAL	other
OPENING	greetings
CLOSING	farewells
THANKING	thanking and responding to thanks

The backward looking function of DAMSL focuses on the relationship of an utterance to previous utterances by the other speaker. These include accepting and rejecting proposals (since DAMSL is focused on task-oriented dialogue), and grounding and repair acts:

Backward Looking Function	
AGREEMENT	speaker's response to previous proposal
ACCEPT	accepting the proposal
ACCEPT-PART	accepting some part of the proposal
MAYBE	neither accepting nor rejecting the proposal
REJECT-PART	rejecting some part of the proposal
REJECT	rejecting the proposal
HOLD	putting off response, usually via subdialogue
ANSWER	answering a question
UNDERSTANDING	whether speaker understood previous
SIGNAL-NON-UNDER.	speaker didn't understand
SIGNAL-UNDER.	speaker did understand
ACK	demonstrated via continuer or assessment
REPEAT-REPHRASE	demonstrated via repetition or reformulation
COMPLETION	demonstrated via collaborative completion

Figure 19.20 shows a labeling of our sample conversation using versions of the DAMSL Forward and Backward tags.

Traum and Hinkelman (1992) proposed that the core speech acts and grounding actts that constitute dialogue acts could fit into an even richer hierarchy of **conversation acts**. Figure 19.21 shows the four levels of act types they propose, with the two levels corresponding to DAMSL dialogue acts as the middle two (grounding and core speech acts). What is new is the idea of turn-taking acts, as well as the kind of coherence relations that we saw in Chapter 18.

CONVERSATION ACTS

[assert]	C_1:	...I need to travel in May.
[info-req,ack]	A_1:	And, what day in May did you want to travel?
[assert, answer]	C_2:	OK uh I need to be there for a meeting that's from the 12th to the 15th.
[info-req,ack]	A_2:	And you're flying into what city?
[assert,answer]	C_3:	Seattle.
[info-req,ack]	A_3:	And what time would you like to leave Pittsburgh?
[check,hold]	C_4:	Uh hmm I don't think there's many options for non-stop.
[accept,ack]	A_4:	Right.
[assert]		There's three non-stops today.
[info-req]	C_5:	What are they?
[assert, open-option]	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.
[accept,ack]	C_6:	OK I'll take the 5ish flight on the night before on the 11th.
[check,ack]	A_6:	On the 11th?
[assert,ack]		OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
[ack]	C_7:	OK.

Figure 19.20 A potential DAMSL labeling of the fragment in Figure 19.4.

Act type	Sample Acts
turn-taking	take-turn, keep-turn, release-turn, assign-turn
grounding	acknowledge, repair, continue
core speech acts	inform, wh-question, accept, request, offer
argumentation	elaborate, summarize, question-answer, clarify

Figure 19.21 Conversation act types, from Traum and Hinkelman (1992).

The acts form a hierarchy, in that performance of an act at a higher level (for example a core speech act) entails performance of a lower level act (taking a turn). We will see the use of conversational acts in generation later on in this section, and will return to the question of coherence and dialogue structure in Section 19.7.

Interpreting Dialogue Acts

How can we do dialogue act interpretation, deciding whether a given input is a QUESTION, a STATEMENT, a SUGGEST (directive), or an ACKNOWLEDGEMENT? Perhaps we can just rely on surface syntax? We saw in Chapter 9 that 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:

(19.21)	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, as is clear from Abbott and Costello's famous *Who's on First* routine at the beginning of the chapter, the mapping from surface form to illocutionary act is complex. For example, the following ATIS utterance looks like a YES-NO-QUESTION meaning something like *Are you capable of giving me a list of...?*:

(19.22) 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 more like *Please give me a list of....* Thus what looks on the surface like a QUESTION can really be a REQUEST.

Similarly, what looks on the surface like a STATEMENT can really be a QUESTION. The very common CHECK question (Carletta et al., 1997b; Labov and Fanshel, 1977), is used to ask 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.

Utterances which use a surface statement to ask a question, or a surface question to issue a request, are called **indirect speech acts**. How can a surface yes-no-question like *Can you give me a list of the flights from Atlanta to Boston?* be mapped into the correct illocutionary act REQUEST? INDIRECT SPEECH ACTS

Dialogue act interpretation can be modeled as a supervised classification task, with dialogue act labels as hidden classes to be detected. Machine-learning classifiers are trained on a corpus in which each utterance is hand-labeled for dialogue acts. The features used in dialogue act interpretation derive from the conversational context and from the act's **microgrammar** (Goodwin, 1996): lexical, collocation, and prosodic features characteristic of the act. Stolcke et al. (2000a), for example, used three kinds of features: MICROGRAMMAR

1. **Words and Collocations:** *Please* or *would you* is a good cue for a REQUEST, *are you* for YES-NO-QUESTIONs.
2. **Prosody:** Rising pitch is a good cue for a YES-NO-QUESTION. Loudness or stress can help distinguish the *yeah* that is an AGREEMENT from the *yeah* that is a BACKCHANNEL.

3. **Conversational Structure:** A *yeah* following a proposal is probably an AGREEMENT; a *yeah* after an INFORM is likely a BACKCHANNEL.

We can integrate these cues into a dialogue act classifier by using an HMM, in which the dialogue acts are the hidden events (Nagata and Morimoto, 1994; Woszczyna and Waibel, 1994; Reithinger et al., 1996; Kita et al., 1996; Warnke et al., 1997; Chu-Carroll, 1998; Stolcke et al., 1998; Taylor et al., 1998; Stolcke et al., 2000b). In the HMM approach, given all available evidence E about a conversation, the goal is to find the dialogue act sequence $D = \{d_1, d_2 \dots, d_N\}$ that has the highest posterior probability $P(D|E)$ given that evidence (as usual here we use capital letters for sequences). Applying Bayes' Rule we get

$$\begin{aligned} D^* &= \operatorname*{argmax}_D P(D|E) \\ &= \operatorname*{argmax}_D \frac{P(D)P(E|D)}{P(E)} \\ &= \operatorname*{argmax}_D P(D)P(E|D) \end{aligned} \tag{19.23}$$

Assuming the three types of evidence (words, prosody, and conversational structure) and making an (incorrect but) simplifying assumption that the prosody and the words are independent, we can estimate the evidence likelihood for a sequence of dialogue acts D as in (19.24):

$$P(E|D) = P(F|D)P(W|D) \tag{19.24}$$

$$D^* = \operatorname*{argmax}_D P(D)P(F|D)P(W|D) \tag{19.25}$$

The resulting equation (19.25) thus has three components, one for each of the kinds of cues discussed above. Let's briefly discuss each of these three components. The prior probability of a sequence of dialogue acts $P(D)$ acts as a model of conversational structure. Drawing on the idea of adjacency pairs (Schegloff, 1968; Sacks et al., 1974) introduced above, we can make the simplifying assumption that conversational structure is modeled as a Markov sequence of dialogue acts.

$$P(D) = \prod_{i=2}^{M} P(d_i|d_{i-1}...d_{i-M+1}) \tag{19.26}$$

Woszczyna and Waibel (1994) give the dialogue HMM shown in Figure 19.22 for a Verbmobil-like appointment scheduling task.

The lexical component of the HMM likelihood, designed to capture the microgrammar of each dialogue act, is modeled by training a separate word-N-gram grammar for each dialogue act, just as we saw with the concept HMM.

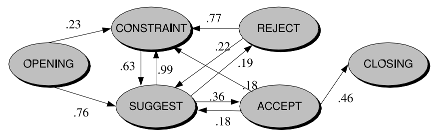

Figure 19.22 A dialogue act HMM (after Woszczyna and Waibel (1994))

$$P(W|D) = \prod_{i=2}^{N} P(w_i|w_{i-1}...w_{i-N+1}, d_i) \tag{19.27}$$

Prosodic models of dialogue act microgrammar rely on accents, boundaries, or their acoustic correlates like F0, duration, and energy. For example the pitch rise at the end of YES-NO-QUESTIONS is a useful cue (Sag and Liberman, 1975; Pierrehumbert, 1980; Waibel, 1988; Daly and Zue, 1992; Kompe et al., 1993; Taylor et al., 1998). Declarative utterances (like STATEMENTS) have **final lowering**: a drop in F0 at the end of the utterance (Pierrehumbert, 1980). FINAL LOWERING

Shriberg et al. (1998) trained CART-style decision trees on simple acoustically-based prosodic features such as the slope of F0 at the end of the utterance, the average energy at different places in the utterance, and various duration measures, normalized in various ways. They found that these features were useful, for example, in distinguishing the four dialogue acts STATEMENT (S), YES-NO QUESTION (QY), DECLARATIVE-QUESTIONS like CHECKS (QD) and WH-QUESTIONS (QW). Figure 19.23 shows the decision tree which gives the posterior probability $P(d|F)$ of a dialogue act d type given sequence of acoustic features F. Note that the difference between S and QY toward the right of the tree is based on the feature `norm_f0_diff` (normalized difference between mean F0 of end and penultimate regions), while the difference between WQ and QD at the bottom left is based on `utt_grad`, which measures F0 slope across the whole utterance.

Decision trees produce a posterior probability $P(d|f)$, and equation (19.25) requires a likelihood $P(F|d)$. Therefore we need to massage the output of the decision tree by Bayesian inversion (dividing by the prior $P(d_i)$ to turn it into a likelihood). If we make the simplifying assumption that the prosodic decisions for each sentence are independent of other sentences, we arrive at the following final equation for HMM tagging of dialogue acts:

$$\begin{aligned} D^* &= \underset{D}{\operatorname{argmax}} P(D)P(F|D)P(W|D) \\ &= \prod_{i=2}^{M} P(d_i|d_{i-1}...d_{i-M+1}) \prod_{i=1}^{N} \frac{P(d_i|F)}{P(d_i)} \prod_{i=2}^{N} P(w_i|w_{i-1}...w_{i-N+1}, d_i) \end{aligned} \tag{19.28}$$

Standard HMM decoding techniques (like Viterbi) can then be used to search for this most-probable sequence of dialogue acts given the sequence of input utterances culminating in the user's most recent utterance.

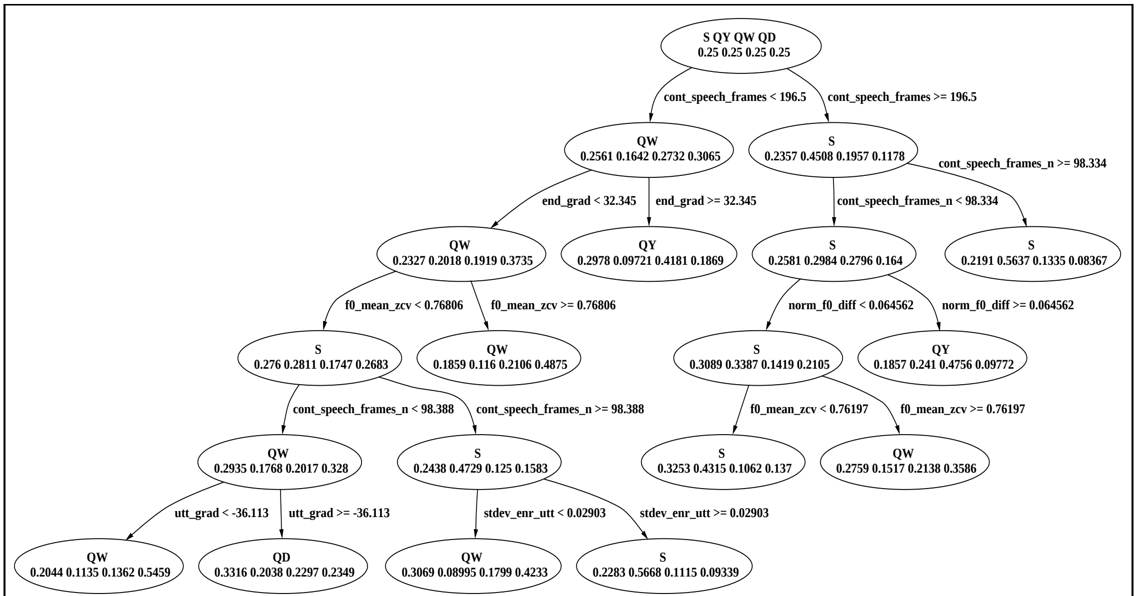

Figure 19.23 Decision tree for the classification of STATEMENT (S), YES-NO QUESTIONS (QY), WH-QUESTIONS (QW) and DECLARATIVE QUESTIONS (QD), after Shriberg et al. (1998). Each node in the tree shows four probabilities, one for each of the four dialogue acts in the order S, QY, QW, QD; the most likely of the four is shown as the label for the node.

Many statistical classifiers besides HMMs have been used to interpret dialogue acts, as well as the non-statistical BDI model, which interprets dialogue acts based on plan-inference (see Section 19.7).

Detecting Correction Acts

In addition to the general-purpose dialogue act interpretation algorithms just described, we may want to build special-purpose detectors for particularly important acts. In this section we briefly introduce the issues in designing one such detector, for the recognition of user **correction** of system errors. If a dialogue system misrecognizes an utterance (usually as a result of ASR errors) the user will generally correct the error by repeating themselves, or rephrasing the utterance. Dialogue systems need to recognize that users are doing a correction, and then figure out what the user is trying to correct, perhaps by interacting with the user further.

CORRECTION

Unfortunately, corrections are actually *harder* to recognize than normal sentences. Swerts et al. (2000) found that corrections in the TOOT dialogue system were misrecognized about twice as often (in terms of WER) as non-corrections. One reason for this is that speakers use a very different prosodic style called **hyperarticulation** for corrections. In hyperarticulated speech, some part of the utterance has exaggerated energy, duration, or F0 contours, such as *I said BAL-TI-MORE, not Boston* (?; Oviatt et al., 1998; Levow, 1998; Hirschberg et al., 2001).

HYPERARTICULATION

Speakers use various strategies to make corrections; Swerts et al. (2000) categorize these as repetition (REP), paraphrase (PAR), adding content (ADD), and omitting content (OMIT), as shown in Figure 19.24. They found that most corrections were repetitions and omissions rather than paraphrases.

	Turn	Corr Type
S	On which day of the week do you want to leave?	
S	Sorry, I didn't hear anything from you. Please tell me your departure date	
U	*Um tomorrow*	NA
S	You can say the departure date. For example, you can say 'today' or 'tomorrow'. Or you can say 'Monday', 'Tuesday', 'Wednesday'...	
U	*Tomorrow*	REP
S	You can say the departure date. For example, you can say 'today' or 'tomorrow'. Or you can say 'Monday', 'Tuesday', 'Wednesday'...	
U	*Saturday*	PAR
S	Do you want to leave on Saturday?	
U	Yes	
S	What time do you want to leave?	
U	Eight o'clock A M	
S	Do you want to leave around ten oclock p-m	
U	*Eight o'clock*	OMIT
S	Do you want to leave around ten oclock p-m	
U	*No*	OMIT
S	What time do you want to leave?	
U	*Eight o'clock A M*	ADD

Figure 19.24 Dialogue fragment from the TOOT system labeled for correction type, from Swerts et al. (2000).

Various machine learning classifiers have been applied to the problem of detecting correction acts (Levow, 1998; Hirschberg et al., 2001; Bulyko et al., 2004). Useful features include lexical information (words like "no", "correction", "I don't", swear words), prosodic and hyperarticulation features (increases in F0 range, pause duration, and word duration, generally normalized by the values

Cue	Turn-taking acts signaled
um	KEEP-TURN, TAKE-TURN, RELEASE-TURN
<lipsmack>, <click>, so, uh	KEEP-TURN, TAKE-TURN
you know, isn't that so	ASSIGN-TURN

Figure 19.25 Language used to perform turn-taking acts, from Stent (2002).

for previous sentences), features indicating utterance length, ASR features (confidence, language model probability), and various dialogue features.

In addition to correction detection, a conversational agent also needs appropriate control or update rules in the dialogue manager (Bulyko et al., 2004).

Generating Dialogue Acts: Confirmation and Rejection

Deciding which dialogue acts to generate is a an extremely complex problem that has received much less attention than the problem of dialogue act interpretation. Stent (2002) is one recent model of dialogue act generation in the TRIPS system (Allen et al., 2001), based on Conversation Acts (page 38) and the BDI model to be described in Section 19.7. Stent uses a set of update rules for content planning. One such rule says that if a user has just released the turn, the system can perform a TAKE-TURN act. Another rule says that if the system has a problem-solving need to summarize some information for the user, then it should use the ASSERT conversation act with that information as the semantic content. The content is then mapped into words using the same techniques as the other natural language generation systems described in Chapter 20. After an utterance is generated, the information state (discourse context) is updated with its words, syntactic structure, semantic form, and semantic and conversation act structure. We will sketch in Section 19.7 some of the issues in modeling and planning that make generation a tough ongoing research effort.

Stent showed that a crucial issue in dialogue generation that doesn't occur in monologue text generation is turn-taking acts. Figure 19.25 shows some example of the turn-taking function of various linguistic forms, from her labeling of conversation acts in the Monroe corpus.

In the rest of this section, rather than trying to cover the entire field, we'll focus narrowly on one kind of dialogue act generation task that is probably the most explored in this relatively new research area. This is the task of generating **confirmation** and **rejection** acts. Because this task is often solved by probabilistic methods, we'll continue this discussion in Section 19.6.

In a dialogue system, mishearings are a particularly important class of problems, because speech recognition has such a high error rate. It is therefore important for dialogue systems to make sure that they have achieved the correct interpretation of the user's input. This is generally done by confirmation with the user. A system using the **explicit confirmation** strategy asks the user a direct question to confirm their understanding. Here are two examples of explicit confirmations from travel planning systems. The (boldface) confirmation questions are both yes-no questions, one using a single sentence, the other presenting a declarative sentence followed by a tag question (a short question phrase like "right?" or "isn't it?"):

EXPLICIT CONFIRMATION

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

A system using the **implicit confirmation** strategy, rather than asking a direct question, uses the *demonstration* or *display* grounding strategies described above, confirming to the user by repeating back what the system understood the user to have said:

IMPLICIT CONFIRMATION

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 the user can just answer 'no' to the confirmation question (Danieli and Gerbino, 1995; Krahmer et al., 1999). But explicit confirmation is awkward and increases the length of the conversation (Danieli and Gerbino, 1995; Walker et al., 1998). The explicit confirmation dialogue fragments above sound non-natural and definitely non-human; implicit confirmation is much more conversationally natural.

While early dialogue systems tended to fix the choice of explicit or implicit confirmation, recent systems treat the question of how to confirm more like a dialogue act generation task, in which the confirmation strategy is adaptive, changing from sentence to sentence.

Various factors can be used in making this decision. The most important factor is some measure of ASR performance. A number of systems, for example, use the acoustic confidence that the ASR system assigns to an utterance, computed from the acoustic log-likelihood of the utterance, to decide whether to make an explicit confirmation. Such systems explicitly confirm sentences for which the recognizer was not confident of its output (Bouwman et al., 1999; San-Segundo et al., 2001; Litman et al., 1999; Litman and Pan, 2002). Recent research has focused on more sophisticated measures of confidence that go beyond acoustic log-likelihood, such as prosodic factors; for example utterances with longer prior pauses, F0 excursions, and longer durations are likely to be misrecognized, (Litman et al., 2000). Another important factor in deciding whether to explicitly confirm is the cost of an error; obviously before actually booking a flight or moving money in an account, explicit confirmation is important (Kamm, 1994; Cohen et al., 2004). All of these factors can thus be combined in a machine learning approach to predict whether explicit confirmation should be used. This can be done with a simple classifier, or via more complex methods; what is required needed is that the information-state include information about utterance prosody, ASR confidence.

Rejection

REJECTION

Confirmation is just one kind of conversational action that a system has to express lack of understanding. Another option is **rejection**. An ASR system rejects an utterance by giving the user a prompt like *I'm sorry, I didn't understand that*, as in the VoiceXML `nomatch` prompts we saw in Section 19.3. Rejection might happen when the ASR confidence is so low, or the best interpretation is so semantically ill-formed, that the system can be relatively sure that the user's input was not recognized at all. Systems thus might have a three-tiered level of confidence; below a certain confidence threshold, an utterance is rejected. Above the threshold, it is explicitly confirmed. If the confidence is even higher, the utterance is implicitly confirmed.

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; ?) 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. 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. 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) summarizes experiments showing that users greatly prefer rapid reprompting as a first-level error prompt.

RAPID REPROMPTING

Instead of rejecting or confirming entire utterances, it would be nice to be able to clarify only the parts of the utterance that the system didn't understand. If a system can assign confidence at a more fine-grained level than the utterance, it can clarify such individual elements via **clarification subdialogues**.

CLARIFICATION SUBDIALOGUES

19.6 MARKOV DECISION PROCESS ARCHITECTURE

One of the fundamental insights of the information-state approach to dialogue architecture is that the choice of conversational actions is dynamically dependent on the current information state. The previous section discussed how dialogue systems could change confirmation and rejection strategies based on context. For example if the ASR or NLU confidence is low, we might choose to do explicit confirmation. If confidence is high, we might chose implicit confirmation, or even decide not to confirm at all. Using a dynamic strategy lets us choose the action which maximizes dialogue success, while minimizing costs. This idea of changing the actions of a dialogue system based on optimizing some kinds of rewards or costs is the fundamental intuition behind modeling dialogue as a **Markov decision process**. This model extends the information-state model by adding a probabilistic way of deciding on the proper actions given the current state.

MARKOV DECISION PROCESS

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 which 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'll need to look at a tutorial example in which the state space is extremely reduced. Thus we'll return to the simple frame-and-slot world, looking at a pedagogical MDP implementation taken from Levin et al. (2000). Their tutorial example is a "Day-and-Month" dialogue system, whose goal is to get correct values of day and month for a two-slot frame via the shortest possible interaction with the user.

MDP

In principle, a state of an MDP could include any possible information about the dialogue, such as the complete dialogue 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 which 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 users 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*. If we assume a special initial state s_i and final state s_f, there are a total of 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), and 31 states with a day but no month (m=0, d=1,2,...31)).

Actions of a MDP dialogue 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 dialogue

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:

$$R = -(w_i n_i + w_e n_e + w_f n_f) \tag{19.29}$$

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 which are filled (0, 1, or 2), and the *w*s are weights.

Finally, a dialogue 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 dialogues shown in Figure 19.26.

In policy 1, the action specified for the no-date/no-month state is to ask for a day, while 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 leads to shorter dialogues but is likely to cause more errors, while 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 Figure 19.26 is thus $-3 \times w_i + 2 \times p_d \times w_e$. The reward for the second dialog in Figure 19.26 is $-2 \times w_i + 2 \times p_d \times w_e$. The directive prompt policy, policy 2, is thus better than policy 1 when the improved error rate justifies the longer interaction, i.e., when $p_o - p_d > \frac{w_i}{2w_e}$.

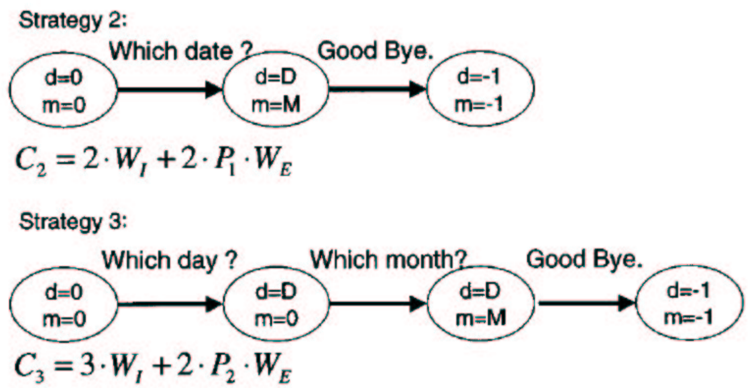

Figure 19.26 REDRAW FIG 2 WITH JUST 2 POLICIES, IN REVERSE ORDER, after Levin et al. (2000).PLACEHOLDER FIGURE.

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 Figure 19.26, we can think of any particular dialogue as a trajectory in state space:

$$s_1 \rightarrow_{a1,r1} s_2 \rightarrow_{a2,r2} s_3 \rightarrow_{a3,r3} \cdots \tag{19.30}$$

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 REWARDS

$$Q([s_0,a_0,s_1,a_1,s_2,a_2\cdots]) = R(s_0,a_0)+\gamma R(s_1,a_1)+\gamma^2 R(s_2,a_2)+\cdots, \tag{19.31}$$

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

$$Q(s,a) = R(s,a)+\gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \tag{19.32}$$

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 once there we take the optimal action a.

Equation (19.32) makes use of two parameters. We need a model of 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 dialogues, 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', to estimate $P(s'|s,a)$. Similarly, if we had a hand-labeled reward for each dialogue, we could build a model of $R(s,a)$.

VALUE ITERATION

Given these parameters, it turns out that there is an iterative algorithm for solving the Bellman equation and determining proper Q values, the **value iteration** algorithm (?). We won't present this here, but see Chapter 17 of Russell and Norvig (2002) for the details of the algorithm as well as further information on Markov Decision Processes.

How do we get enough labeled training data to set these parameters? This is especially worrisome in any real problem, where the number of states s is extremely large. Two methods have been applied in the past. The first is to carefully hand-tune the states and policies so that there are a very small number of states and policies that need to be set automatically. In this case we can build a dialogue system which explore the state space by generating random conversations. Probabilities can then be set from this corpus of conversations. The second 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.

The random conversation approach was taken by Singh et al. (2002). They used reinforcement learning to make a small set of optimal policy decisions. Their NJFun system learned to choose actions which varied the initiative (system, user, or mixed) and the confirmation strategy (explicit or none). The state of the system was specified by values of 7 features including which slot in the frame is being worked on (1-4), the ASR confidence value (0-5), how many times a current slot question had been asked, whether a restrictive or non-restrictive grammar was used, and so on. The result of using only 7 features with a small number of attributes resulted in a small state space (62 states). Each state had only 2 possible actions (system versus user initiative when asking questions, explicit versus no confirmation when receiving answers). They ran the system with real users, creating 311 conversations. Each conversation had a very simple binary reward function; 1 if the user completed the task (finding specified museums, theater, winetasting in the New Jersey area), 0 if the user did not. The system successful learned a good dialogue policy (roughly, start with user initiative, then back of to either mixed or system initiative when reasking for an attribute; confirm only at lower confidence values; both initiative and confirmation policies, however, are different for different attributes). They showed that their policy actually was more successful based on various objective measures than many hand-designed policies reported in the literature.

The simulated user strategy was taken by Levin et al. (2000), in their MDP model with reinforcement learning in the ATIS task. Their simulated user was a generative stochastic model that given the system's current state and actions, produces a frame-slot representation of a user response. The parameters of the simulated user were estimated from a corpus of ATIS dialogues. The simulated user was then used to interact with the system for tens of thousands of conversations, leading to an optimal dialogue policy. Simulation can also be run in reverse; (Williams and Young, 2003) used a Wizard-of-Oz simulation of an MDP dialogue system to help determine the state space and action set and learn the initial policy.

While the MDP architecture offers a powerful new way of modeling dialogue behavior, it relies on the problematic assumption that the system actually knows what state it is in. This is of course not true in a number of ways; the system never knows the true internal state of the user, and even the state in the dialogue may be obscured by speech recognition errors. Recent attempts to relax this assumption have relied on Partially Observable Markov Decision Processes, or POMDPs (sometimes pronounced 'pom-deepeez'). In a POMPDB, we model the user output as an observed signal generated from yet another hidden variable. See Roy et al. (2000), Young (2002), and Russell and Norvig (2002).

19.7 ADVANCED: PLAN-BASED DIALOGUE AGENTS

One of the earliest models of conversational agent behavior, and also one of the most sophisticated, is based on the use of AI planning techniques. For example, the Rochester TRIPS agent (Allen et al., 2001) simulates helping with emergency management, planning where and how to supply ambulances or personnel in a simulated emergency situation. The same planning algorithms that reason how to get an ambulance from point A to point B can be applied to conversation as well. Since communication and conversation are just special cases of rational action in the world, these actions can be planned like any other. So 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 what plan the interlocutor might have had to cause them to say what they said.

Using plans to generate and interpret sentences in this way require that the planner have good models of its **beliefs**, **desires**, and **intentions** (BDI), as well as

BDI

those of the interlocutor. Plan-based models of dialogue are thus often referred to as **BDI** models. BDI models of dialogue were first introduced by Allen, Cohen, Perrault, and their colleagues and students 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. In another related line of research, Grosz and her colleagues and students showed how using similar notions of intention and plans allowed ideas of discourse structure and coherence to be applied to dialogue.

Plan-Inferential Interpretation and Production

Let's first sketch out the ideas of plan-based comprehension and production. How might a plan-based agent act as the human travel agent to understand sentence C_2 in the dialogue repeated below?

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.

The Gricean principle of Relevance can be used to infer that the client's meeting is relevant to the flight booking. The system may know that one precondition for having a meeting (at least before web conferencing) is being at the place where the meeting is in. One way of being at a place is flying there, and booking a flight is a precondition for flying there. The system can follow this chain of inference, abducing that user wants to fly on a date before the 12th.

Next, consider how our plan-based agent could act as the human travel agent to produce sentence A_1 in the dialogue above. The planning agent would reason that in order to help a client book a flight it must know enough information about the flight to book it. It reasons that knowing the month (May) is insufficient information to specify a departure or return date. The simplest way to find out the needed date information is to ask the client.

In the rest of this section, we'll flesh out the sketchy outlines of planning for understanding and generation using Perrault and Allen's formal definitions of belief and desire in the predicate calculus. Reasoning about belief is done with a number of axiom schemas inspired by Hintikka (1969). We'll represent "S believes the proposition P" as the two-place predicate $B(S,P)$, with axioms such as $B(A,P) \wedge B(A,Q) \Rightarrow B(A,P \wedge Q)$. Knowledge is defined as "true belief"; *S knows that P* will be represented as $KNOW(S,P)$, defined as $\text{KNOW}(S,P) \equiv P \wedge B(S,P)$.

The theory of desire relies on the predicate WANT. If an agent S wants P to be true, we say $WANT(S,P)$, or $W(S,P)$ for short. P can be a state or the execution of some action. Thus if ACT is the name of an action, $W(S, \mathrm{ACT}(H))$ means that S wants H to do ACT. The logic of WANT relies on its own set of axiom schemas just like the logic of belief.

The BDI models also require an axiomatization of actions and planning; the simplest of these is based on a set of **action schema**s based on the simple AI planning model STRIPS (Fikes and Nilsson, 1971). Each action schema has a set of parameters with *constraints* about the type of each variable, and three parts:

ACTION SCHEMA

- *Preconditions:* Conditions that must already be true to perform the action.
- *Effects:* Conditions that become true as a result of performing the action.
- *Body:* A set of partially ordered goal states that must be achieved in performing the action.

In the travel domain, for example, the action of agent A booking flight $F1$ for client C might have the following simplified definition:

BOOK-FLIGHT(A,C,F):
Constraints: Agent(A) ∧ Flight(F) ∧ Client(C)
Precondition: Know(A,depart-date(F)) ∧ Know(A,depart-time(F)) ∧ Know(A,origin(F)) ∧ Know(A,flight-type(F)) ∧ Know(A,destination(F)) ∧ Has-Seats(F) ∧ W(C,(BOOK(A,C,F))) ∧ . . .
Effect: Flight-Booked(A,C,F)
Body: Make-Reservation(A,F,C)

This same kind of STRIPS action specification can be used for speech acts. INFORM is the speech act of informing the hearer of some proposition, based on Grice's (1957) idea that a speaker informs the hearer of something merely by causing the hearer to believe that the speaker wants them to know something:

INFORM(S,H,P):
Constraints: Speaker(S) ∧ Hearer(H) ∧ Proposition(P)
Precondition: Know(S,P) ∧ W(S, INFORM(S, H, P))
Effect: Know(H,P)
Body: B(H,W(S,Know(H,P)))

REQUEST is the directive speech act for requesting the hearer to perform some action:

REQUEST(S,H,ACT):
Constraints: Speaker(S) ∧ Hearer(H) ∧ ACT(A) ∧ H is agent of ACT
Precondition: W(S,ACT(H))
Effect: W(H,ACT(H))
Body: B(H,W(S,ACT(H)))

Let's now see how a plan-based dialogue system might interpret the sentence:

C_2: I need to be there for a meeting that's from the 12th to the 15th.

We'll assume the system has the BOOK-FLIGHT plan mentioned above. In addition, we'll need knowledge about meetings and getting to them, in the form of the MEETING, FLY-TO, and TAKE-FLIGHT plans, sketched broadly below:

MEETING(P,L,T1,T2):
Constraints: Person(P) $\wedge$ Location (L) $\wedge$ Time (T1) $\wedge$ Time (T2) $\wedge$ Time (TA)
Precondition: At (P, L, TA)
Before (TA, T1)
Body: ...

FLY-TO(P, L, T):
Constraints: Person(P) $\wedge$ Location (L) $\wedge$ Time (T)
Effect: At (P, L, T)
Body: TAKE-FLIGHT(P, L, T)

TAKE-FLIGHT(P, L, T):
Constraints: Person(P) $\wedge$ Location (L) $\wedge$ Time (T) $\wedge$ Flight (F) $\wedge$ Agent (A)
Precondition: BOOK-FLIGHT (A, P, F)
Destination-Time(F) = T
Destination-Location(F) = L
Body: ...

Now let's assume that an NLU module returns a semantics for the client's utterance which (among other things) includes the following semantic content:

MEETING (P, ?L, T1, T2)
Constraints: P = Client $\wedge$ T1 = May 12 $\wedge$ T2 = May 15

Our plan-based system now has two plans established, one MEETING plan from this utterance, and one BOOK-FLIGHT plan from the previous utterance. The system implicitly uses the Gricean Relevance intuition to try to connect them. Since BOOK-FLIGHT is a precondition for TAKE-FLIGHT, the system may hypothesize (infer) that the user is planning a TAKE-FLIGHT. Since TAKE-FLIGHT is in the body of FLY-TO, the system further infers a FLY-TO plan. Finally, since the effect of FLY-TO is a precondition of the MEETING, the system can unify each of the people, locations, and times of all of these plans. The result will be that the system knows that the client wants to arrive at the destination before May 12th.

Let's turn to the details of our second example:

C_1: I need to travel in May.
A_1: And, what day in May did you want to travel?

How does a plan-based agent know to ask question A_1? This knowledge comes from the BOOK-FLIGHT plan, whose preconditions were that the agent know a variety of flight parameters including the departure date and time, origin and destination cities, and so forth. Utterance C_1 contains the origin city and partial information about the departure date; the agent has to request the rest. A plan-based agent would use an action schema like REQUEST-INFO to represent a plan for asking information questions (simplified from Cohen and Perrault (1979)):

REQUEST-INFO(A,C,I):

Constraints:	Agent(A) $\wedge$ Client(C)
Precondition:	Know(C,I)
Effect:	Know(A,I)
Body:	B(C,W(A,Know(A,I)))

Because the effects of REQUEST-INFO match each precondition of BOOK-FLIGHT, the agent can use REQUEST-INFO to achieve the missing information.

Dialogue Structure and Coherence

Section **??** described an approach to dialogue structure based on a set of coherence relations. In order to determine that a coherence relation holds, the system must reason about the constraints that the relation imposes on the **information** in the utterances. This *informational* approach to coherence has been applied predominantly to monologues. The BDI approach to utterance interpretation gives rise to another view of coherence, which we will call the **intentional** approach or **intentional structure**. According to this approach, the hearer must infer the plan-based intentions of the speaker underlying each utterance. INTENTIONAL STRUCTURE

The fundamental idea in this model, due to Grosz and Sidner (1986), is hence that a discourse has associated with it an underlying purpose that is held by the person who initiates it, called the **discourse purpose** (DP). Each discourse segment within the discourse has a corresponding purpose, a **discourse segment purpose** (DSP), which has a role in achieving the overall DP. Possible DPs/DSPs include: DISCOURSE PURPOSE DISCOURSE SEGMENT PURPOSE

1. Intend that some agent intend to perform some physical task.
2. Intend that some agent believe some fact.
3. Intend that some agent believe that one fact supports another.
4. Intend that some agent know some property of an object.

As opposed to the larger sets of coherence relations used in informational accounts of coherence, Grosz and Sidner propose only two such relations: **dominance** and **satisfaction-precedence**. DSP_1 dominates DSP_2 if satisfying DSP_2 is intended to provide part of the satisfaction of DSP_1. DSP_1 satisfaction-precedes DSP_2 if DSP_1 must be satisfied before DSP_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 19.27 A fragment from a telephone conversation between a client (C) and a travel agent (A) (repeated from Figure 19.4).

Consider the dialogue between a client (C) and a travel agent (A) that we saw earlier, repeated here in Figure 19.27. Collaboratively, the caller and agent successfully identify a flight that suits the caller's needs. Achieving this joint goal requires that a top-level discourse intention be satisfied, listed as I1 below, in addition to several intermediate intentions that contributed to the satisfaction of I1, listed as I2-I5:

I1: (Intend C (Intend A (A find a flight for C)))

I2: (Intend A (Intend C (Tell C A departure date)))

I3: (Intend A (Intend C (Tell C A destination city)))

I4: (Intend A (Intend C (Tell C A departure time)))

I5: (Intend C (Intend A (A find a nonstop flight for C)))

Intentions I2–I5 are all subordinate to intention I1, as they were all adopted to meet preconditions for achieving intention I1. This is reflected in the dominance relationships below:

I1 dominates I2 $\wedge$ I1 dominates I3 $\wedge$ I1 dominates I4 $\wedge$ I1 dominates I5

Furthermore, intentions I2 and I3 needed to be satisfied before intention I5, since the agent needed to know the departure date and destination in order to start listing nonstop flights. This is reflected in the satisfaction-precedence relationships below:

I2 satisfaction-precedes I5 $\wedge$ I3 satisfaction-precedes I5

The dominance relations give rise to the discourse structure depicted in Figure 19.28. Each discourse segment is numbered in correspondence with the intention number that serves as its DP/DSP.

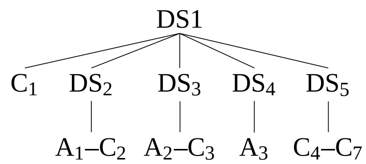

Figure 19.28 Discourse Structure of the Flight Reservation Dialogue

Intentions and their relationships give rise to a coherent discourse based on their role in the overall *plan* that the caller is inferred to have. We assume that the caller and agent have the plan BOOK-FLIGHT described on page 53. This plan requires that the agent know the departure time and date and so on. As we discussed above, the agent can use the REQUEST-INFO action scheme from page 55 to ask the user for this information.

Discourse segments DS2 and DS3 are cases in which performing REQUEST-INFO succeeds for identifying the values of the departure date and destination city parameters respectively. Segment DS4 is also a request for a parameter value (departure time), but is unsuccessful in that the caller takes the initiative instead, by (implicitly) asking about nonstop flights. Segment DS5 leads to the satisfaction of the top-level DP from the caller's selection of a nonstop flight from a short list that the agent produced.

Subsidiary discourse segments like DS2 and DS3 are also called **subdialogues**. DS2 and DS3 are **knowledge precondition** subdialogues (Lochbaum et al., 1990; Lochbaum, 1998), since they are initiated by the agent to help satisfy preconditions of a higher-level goal (in this case addressing the client's request for travel in May). They are also called **information-sharing subdialogues** (Chu-Carroll and Carberry, 1998).

SUBDIALOGUES

INFORMATION-SHARING SUBDIALOGUES

Algorithms for inferring intentional structure in dialogue work similarly to algorithms for inferring dialogue acts. Many algorithms apply variants of the BDI model (e.g., Litman, 1985; Grosz and Sidner, 1986; Litman and Allen, 1987; Carberry, 1990; Passonneau and Litman, 1993; Chu-Carroll and Carberry, 1998). Machine-learning algorithms rely on features like cue words and phrases (Reichman, 1985; Grosz and Sidner, 1986; Hirschberg and Litman, 1993) or prosody (Grosz and Hirschberg, 1992; Hirschberg and Pierrehumbert, 1986; Hirschberg and Nakatani, 1996), and other cues. For example boundary tones may be used to suggest a dominance relation between two intonational phrases (Pierrehumbert and Hirschberg, 1990)

What is the relationship between informational and intentional coherence? The key to intentional coherence lies in the ability of the interlocutors to recognize each other's intentions and plans. As we saw in the previous chapter, informational coherence lies in the ability to establish content-bearing relationships between utterances. Moore and Pollack (1992), among others, have argued that both levels of analysis must co-exist; for example a speaker can have an intention to motivate her hearer to do something by giving an explanation. The result of this intention is two sentences which may be linked by an Explanation relation.

19.8 ADVANCED: PROCESSING HUMAN-HUMAN DIALOG

In addition to work on building conversational agents, computational dialogue work also focuses on human-human dialogue. We need to process human-human dialogue in order to automatically transcribe or summarize business meetings, to close-caption TV shows, or to building personal telephone assistants that can take notes on telephone conversations.

SEGMENTATION

A key task in human-human conversation is utterance boundary **segmentation**, the task of separating out utterances from each other. Rhis is an important task since many computational dialogue models are based on extracting an utterance as a primitive unit. The segmentation problem is difficult because a single utterance may be spread over several turns (as in (19.33)), or a single turn may include several utterances (as in (19.34)).

(19.33)

A: Yeah um let me see here we've got you on American flight nine thirty eight
C: Yep.
A: leaving on the twentieth of June out of Orange County John Wayne Airport at seven thirty p.m.
C: Seven thirty.
A: and into uh San Francisco at eight fifty seven.

(19.34)

A: Three two three and seven five one. OK and then does he know there is a nonstop that goes from Dulles to San Francisco? Instead of connection through St. Louis.

Segmentation algorithms use boundary **cues** such as:

CUE WORDS

- **cue words:** Cue words like *well*, *and*, *so*, that tend to occur at beginnings and ends of utterances (Reichman, 1985; Hirschberg and Litman, 1993).
- ***N*-gram word or POS sequences:** Specific word or POS sequences that often indicate boundaries. *N*-gram grammars can be trained on a training set

labeled with special utterance-boundary tags. (Mast et al., 1996; Meteer and Iyer, 1996; Stolcke and Shriberg, 1996; Heeman and Allen, 1999).

- **prosody:** Utterance-final prosodic features like boundary tones, phrase-final lengthening and pause duration
- **gaze:** In face-to-face dialogue, **gaze** is an important cue. GAZE

A related task in human-human dialog is **diarization**: assigning each utterance to the talker who produced it; this can be quite hard in multi-speaker meetings. DIARIZATION

19.9 Summary

Conversational agents are a crucial speech and language processing application that are already widely used commercially. Research on these agents relies crucially on an understanding of human dialogue or conversational practices.

- Dialogue systems generally have 5 components: speech recognition, natural language understanding, dialogue management, natural language generation, and speech synthesis. They may also have a task manager specific to the task domain.
- Dialogue architectures for conversational agents include finite-state systems, **frame-based** production systems, and advanced systems such as information-state, Markov Decision Processes, and **BDI** (**belief-desire-intention**) models.
- Turn-taking, grounding, conversational structure, implicature, and initiative are crucial human dialogue phenomena that must also be dealt with in conversational agents.
- Speaking in dialogue is a kind of action; these acts are referred to as speech acts or **dialogue acts**. Models exist for generating and interpreting these acts.
- Human-human dialogue is another important area of dialogue, relevant especially for such computational tasks as **automatic meeting summarization**.

Bibliographical and Historical Notes

Early work on speech and language processing had very little emphasis on the study of dialogue. The dialogue manager for the simulation of the paranoid agent PARRY (Colby et al., 1971), was a little more complex. Like ELIZA, it was based on a production system, but where ELIZA's rules were based only on the words in the user's previous sentence, PARRY's rules also rely on global variables indicating its emotional state. Furthermore, PARRY's output sometimes makes use of script-like sequences of statements when the conversation turns to its delusions. For example, if PARRY's **anger** variable is high, he will choose from a set of "hostile" outputs. If the input mentions his delusion topic, he will increase the value of his **fear** variable and then begin to express the sequence of statements related to his delusion.

The appearance of more sophisticated dialogue managers awaited the better understanding of human-human dialogue. Studies of the properties of human-human dialogue began to accumulate in the 1970's and 1980's. The Conversation Analysis community (Sacks et al., 1974; Jefferson, 1984; Schegloff, 1982) began to study the interactional properties of conversation. Grosz's (1977) dissertation significantly influenced the computational study of dialogue with its introduction of the study of dialogue structure, with its finding that "task-oriented dialogues have a structure that closely parallels the structure of the task being performed" (p. 27), which led to her work on intentional and attentional structure with Sidner. Lochbaum et al. (2000) is a good recent summary of the role of intentional structure in dialogue. The BDI model integrating earlier AI planning work (Fikes and Nilsson, 1971) with speech act theory (Austin, 1962; Gordon and Lakoff, 1971; Searle, 1975a) was first worked out by Cohen and Perrault (1979), showing how speech acts could be generated, and Perrault and Allen (1980) and Allen and Perrault (1980), applying the approach to speech-act interpretation. Simultaneous work on a plan-based model of understanding was developed by (Wilensky, 1983) in the Schankian tradition.

Models of dialogue as collaborative behavior were introduced in the late 1980's and 1990's, including the ideas of common ground (?), reference as a collaborative process (Clark and Wilkes-Gibbs, 1986), and models of **joint intentions** (Levesque et al., 1990), and **shared plans** (Grosz and Sidner, 1980). Related to this area is the study of **initiative** in dialogue, studying how the dialogue control shifts between participants (Walker and Whittaker, 1990; Smith and Gordon, 1997; Chu-Carroll and Brown, 1997).

Work on dialogue acts and dialogue moves drew from the a number of sources, including HCRC's Map Task (Carletta et al., 1997b), from James Allen and his colleagues and students, including Hinkelman and Allen (1989), who showed how lexical and phrasal cues could be integrated into BDI model of speech acts, and Traum (2000), Traum and Hinkelman (1992), and from ? (?)

A wide body of dialogue research came out of AT&T and Bell Laboratories centered around the 1990's, including FIX LIST HERE.

To add: Communicator/atis history, VoiceXML, commercial deployments.

Good surveys on dialogue systems include Sadek and De Mori (1998), McTear (2002, 2004), and the dialogue chapter in Allen (1995).

EXERCISES

19.1 List the dialogue act misinterpretations in the *Who's On First* routine at the beginning of the chapter.

19.2 Write a finite-state automaton for a dialogue manager for checking your bank balance and withdrawing money at an automated teller machine.

19.3 Dispreferred responses (for example turning down a request) are usually signaled by surface cues, such as significant silence. Try to notice the next time you or someone else utters a dispreferred response, and write down the utterance. What are some other cues in the response that a system might use to detect a dispreferred response? Consider non-verbal cues like eye-gaze and body gestures.

19.4 When asked a question to which they aren't sure they know the answer, people display their lack of confidence via cues that resemble other dispreferred responses. Try to notice some unsure answers to questions. What are some of the cues? If you have trouble doing this, read Smith and Clark (1993) and listen specifically for the cues they mention.

19.5 Build a VoiceXML dialogue system for giving the current time around the world. The system should ask the user for a city and a time format (24 hour, etc) and should return the current time, properly dealing with time zones.

19.6 Implement a small air-travel help system based on text input. Your system should get constraints from the user about a particular flight that they want to take, expressed in natural language, and display possible flights on a screen. Make simplifying assumptions. You may build in a simple flight database or you may use a flight information system on the web as your backend.

19.7 Augment your previous system to work with speech input via VoiceXML. (or alternatively, describe the user interface changes you would have to make for it to work via speech over the phone). What were the major differences?

19.8 Design a simple dialogue system for checking your email over the telephone. Implement in VoiceXML.

19.9 Test your email-reading system on some potential users. Choose some of the metrics described in Section 19.4 and evaluate your system.

Allen, J. (1995). *Natural Language Understanding (2nd ed.)*. Benjamin Cummings, Menlo Park, CA.

Allen, J. and Core, M. (1997). Draft of DAMSL: Dialog act markup in several layers. Unpublished manuscript.

Allen, J., Ferguson, G., and Stent, A. (2001). An architecture for more realistic conversational systems. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, Santa Fe, New Mexico, United States, pp. 1–8. ACM Press.

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

Allwood, J. (1995). An activity-based approach to pragmatics. *Gothenburg Papers in Theoretical Linguistics*, *76*.

Allwood, J., Nivre, J., and Ahlsén, E. (1992). On the semantics and pragmatics of linguistic feedback. *Journal of Semantics*, *9*, 1–26.

Atkinson, M. and Drew, P. (1979). *Order in Court*. Macmillan, London.

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

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). GUS, A frame driven dialog system. *Artificial Intelligence*, *8*, 155–173.

Bouwman, G., Sturm, J., and Boves, L. (1999). Incorporating confidence measure in the Dutch Train Timetable Information system developed in the ARISE project. In *IEEE ICASSP-99*, pp. 493–496. IEEE.

Bulyko, I., Kirchhoff, K., Ostendorf, M., and Goldberg, J. (2004). Error-sensitive response generation in a spoken langugage dialogue system. To appear in Speech Communication.

Bunt, H. (1994). Context and dialogue control. *Think*, *3*, 19–31.

Bunt, H. (2000). Dynamic interpretation and dialogue theory, volume 2. In Taylor, M. M., Neel, F., and Bouwhuis, D. G. (Eds.), *The structure of multimodal dialogue*, pp. 139–166. John Benjamins, Amsterdam.

Carberry, S. (1990). *Plan Recognition in Natural Language Dialog*. MIT Press, Cambridge, MA.

Carletta, J., Dahlbäck, N., Reithinger, N., and Walker, M. A. (1997a). Standards for dialogue coding in natural language processing. Tech. rep. Report no. 167, Dagstuhl Seminars. Report from Dagstuhl seminar number 9706.

Carletta, J., Isard, A., Isard, S., Kowtko, J. C., Doherty-Sneddon, G., and Anderson, A. H. (1997b). 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, Menlo Park, CA.

Chu-Carroll, J. and Brown, M. K. (1997). Tracking initiative in collaborative dialogue interactions. In *ACL/EACL-97*, Madrid, Spain, pp. 262–270. Association for Computational Linguistics.

Chu-Carroll, J. and Carberry, S. (1998). Collaborative response generation in planning dialogues. *Computational Linguistics*, *24*(3), 355–400.

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

Chung, G. (2004). Developing a flexible spoken dialog system using simulation. In *Proceedings of ACL-04*, Barcelona, Spain. Association for Computational Linguistics.

Clark, H. H. (1994). Discourse in production. In Gernsbacher, M. A. (Ed.), *Handbook of Psycholinguistics*. Academic Press, San Diego.

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

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

Cohen, P. and Oviatt, S. (1995). The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*, *92*(22), 9921–9927.

Colby, K. M., Weber, S., and Hilf, F. D. (1971). Artificial paranoia. *Artificial Intelligence*, *2*(1), 1–25.

Cole, R. A., Novick, D. G., Vermeulen, P. J. E., Sutton, S., Fanty, M., Wessels, L. F. A., de Villiers, J. H., Schalkwyk, J., Hansen, B., and Burnett, D. (1997). Experiments with a spoken dialogue system for taking the US census. *Speech Communication*, *23*, 243–260.

Cole, R. A., Novick, D. G., Burnett, D., Hansen, B., Sutton, S., and Fanty, M. (1994). Towards automatic collection of the U.S. census. In *IEEE ICASSP-94*, Adelaide, Australia, Vol. I, pp. 93–96. IEEE.

Cole, R. A., Novick, D. G., Fanty, M., Sutton, S., Hansen, B., and Burnett, D. (1993). Rapid prototyping of spoken language systems: The Year 2000 Census

Project. In *Proceedings of the International Symposium on Spoken Dialogue*, Waseda University, Tokyo, Japan.

Core, M., Ishizaki, M., Moore, J. D., Nakatani, C., Reithinger, N., Traum, D., and Tutiya, S. (1999). The report of the third workshop of the Discourse Resource Initiative, Chiba University and Kazusa Academia Hall. Tech. rep. No.3 CC-TR-99-1, Chiba Corpus Project, Chiba, Japan.

Daly, N. A. and Zue, V. W. (1992). Statistical and linguistic analyses of F_0 in read and spontaneous speech. In *ICSLP-92*, Vol. 1, pp. 763–766.

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, Menlo Park, CA.

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.

Fraser, N. M. and Gilbert, G. N. (1991). Simulating speech systems. *Computer Speech and Language*, *5*, 81–99.

Good, M. D., Whiteside, J. A., Wixon, D. R., and Jones, S. J. (1984). Building a user-derived interface. *Communications of the ACM*, *27*(10), 1032–1043.

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, Cambridge.

Gordon, D. and Lakoff, G. (1971). Conversational postulates. In *CLS-71*, pp. 200–213. University of Chicago. Reprinted in Peter Cole and Jerry L. Morgan (Eds.), *Speech Acts: Syntax and Semantics Volume 3*, Academic, 1975.

Gorin, A., Riccardi, G., and Wright, J. (1997). How may i help you?. *Speech Communication*, *23*, 113–127.

Gould, J. D., Conti, J., and Hovanyecz, T. (1983). Composing letters with a simulated listening typewriter. *Communications of the ACM*, *26*(4), 295–308.

Gould, J. D. and Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM*, *28*(3), 300–311.

Grice, H. P. (1957). Meaning. *Philosophical Review*, *67*, 377–388. Reprinted in *Semantics*, edited by Danny D. Steinberg & Leon A. Jakobovits (1971), Cambridge University Press, pages 53–59.

Grice, H. P. (1975). Logic and conversation. In Cole, P. and Morgan, J. L. (Eds.), *Speech Acts: Syntax and Semantics Volume 3*, pp. 41–58. Academic Press, New York.

Grice, H. P. (1978). Further notes on logic and conversation. In Cole, P. (Ed.), *Pragmatics: Syntax and Semantics Volume 9*, pp. 113–127. Academic Press, New York.

Grosz, B. and Hirschberg, J. (1992). Some intonational characteristics of discourse structure. In *ICSLP-92*, Vol. 1, pp. 429–432.

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, Cambridge, MA.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, *12*(3), 175–204.

Guindon, R. (1988). A multidisciplinary perspective on dialogue structure in user-advisor dialogues. In Guindon, R. (Ed.), *Cognitive Science And Its Applications For Human-Computer Interaction*, pp. 163–200. Lawrence Erlbaum, Hillsdale, NJ.

Heeman, P. A. and Allen, J. (1999). Speech repairs, intonational phrases and discourse markers: Modeling speakers' utterances in spoken dialog. *Computational Linguistics*, *25*(4).

Hemphill, C. T., Godfrey, J., and Doddington, G. R. (1990). The ATIS spoken language systems pilot corpus. In *Proceedings DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, pp. 96–101. Morgan Kaufmann.

Hinkelman, E. A. and Allen, J. (1989). Two constraints on speech act ambiguity. In *Proceedings of the 27th ACL*, Vancouver, Canada, pp. 212–219. ACL.

Hintikka, J. (1969). Semantics for propositional attitudes. In Davis, J. W., Hockney, D. J., and Wilson, W. K. (Eds.), *Philosophical Logic*, pp. 21–45. D. Reidel, Dordrecht, Holland.

Hirschberg, J. and Litman, D. J. (1993). Empirical studies on the disambiguation of cue phrases. *Computational Linguistics*, *19*(3), 501–530.

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

Hirschberg, J. and Nakatani, C. (1996). A prosodic analysis of discourse segments in direction-giving monologues. In *Proceedings of ACL-96*, Santa Cruz, CA, pp. 286–293. ACL.

Hirschberg, J. and Pierrehumbert, J. (1986). The intonational structuring of discourse. In *ACL-86*, New York, pp. 136–144. ACL.

Hirschman, L. and Pao, C. (1993). The cost of errors in a spoken language system. In *EUROSPEECH-93*, pp. 1419–1422.

Issar, S. and Ward, W. (1993). Cmu's robust spoken language understanding system. In *Eurospeech 93*, pp. 2147–2150.

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

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

Kamm, C. A. (1994). User interfaces for voice applications. In Roe, D. B. and Wilpon, J. G. (Eds.), *Voice Communication Between Humans and Machines*, pp. 422–442. National Academy Press, Washington, D.C.

Kita, K., Fukui, Y., Nagata, M., and Morimoto, T. (1996). Automatic acquisition of probabilistic dialogue models. In *ICSLP-96*, Philadelphia, PA, Vol. 1, pp. 196–199.

Kompe, R., Kießling, A., Kuhn, T., Mast, M., Niemann, H., Nöth, E., Ott, K., and Batliner, A. (1993). Prosody takes over: A prosodically guided dialog system. In *EUROSPEECH-93*, Berlin, Vol. 3, pp. 2003–2006.

Krahmer, E., Swerts, M., Theune, M., and Weegels, M. (1999). Error spotting in human-machine interactions. In *EUROSPEECH-99*, Budapest, pp. 1423–1426.

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

Landauer, T. K. (Ed.). (1995). *The Trouble With Computers: Usefulness, Usability, and Productivity*. MIT Press, Cambridge, MA.

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). *Pragmatics*. Cambridge University Press, Cambridge.

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

Litman, D. J. (1985). *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. Ph.D. thesis, University of Rochester, Rochester, NY.

Litman, D. J. and Allen, J. F. (1987). A plan recognition model for subdialogues in conversation. *Cognitive Science*, *11*, 163–200.

Litman, D. J., Hircshberg, J. B., and Swerts, M. (2000). Predicting automatic speech recognition performance using prosodic cues. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, Seattle, Washington, pp. 218–225.

Litman, D. J. and Pan, S. (2002). Designing and evaluating an adaptive spoken dialogue system. *User Modeling and User-Adapted Interaction*, *12*(2-3), 111–137.

Litman, D. J. and Silliman, S. (2004). Itspoke: An intelligent tutoring spoken dialogue system. In *Proceedings of HLT/NAACL-2004*.

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

Lochbaum, K. E. (1998). A collaborative planning model of intentional structure. *Computational Linguistics*, *24*(4), 525–572.

Lochbaum, K. E., Grosz, B. J., and Sidner, C. L. (1990). Models of plans to support communication: An initial report. In *AAAI-90*, Boston, MA, pp. 485–490. Morgan Kaufmann.

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

Mast, M., Kompe, R., Harbeck, S., Kießling, A., Niemann, H., Nöth, E., Schukat-Talamazzini, E. G., and Warnke, V. (1996). Dialog act classification with the help of prosody. In *ICSLP-96*, Philadelphia, PA, Vol. 3, pp. 1732–1735.

McTear, M. F. (2002). Spoken dialogue technology: Enabling the conversational interface. *ACM Computing Surveys*, *34*(1), 90–169.

McTear, M. F. (2004). *Spoken Dialogue Technology*. Springer Verlag, London.

Meteer, M. and Iyer, R. (1996). Modeling conversational speech for speech recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, pp. 33–47. ACL.

Miller, S., Bobrow, R., Ingria, R., and Schwartz, R. (1994). Hidden understanding models of natural language. In *Proceedings of the 32nd ACL*, Las Cruces, NM, pp. 25–32. ACL.

Miller, S., Fox, H., Ramshaw, L., and Weischedel, R. (2000). A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL)*, Seattle, Washington, pp. 226–233.

Miller, S., Stallard, D., Bobrow, R., and Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the ACL*, Santa Cruz, California, pp. 55–61.

Moore, J. D. and Pollack, M. E. (1992). A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics*, *18*(4), 537–544.

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.

Nielsen, J. (1992). The usability engineering life cycle. *IEEE Computer*, 12–22.

Norman, D. A. (1988). *The Design of Everyday Things*. Doubleday, New York.

Oviatt, S., Cohen, P. R., Wang, M. Q., and Gaston, J. (1993). A simulation-based research strategy for designing complex NL sysems. In *Proceedings DARPA*

Speech and Natural Language Workshop, Princeton, NJ, pp. 370–375. Morgan Kaufmann.

Oviatt, S., MacEachern, M., and Levow, G.-A. (1998). Predicting hyperarticulate speech during human-computer error resolution. *Speech Communication*, *24*, 87–110.

Passonneau, R. and Litman, D. J. (1993). Intention-based segmentation: Human reliability and correlation with linguistic cues. In *Proceedings of the 31st ACL*, Columbus, Ohio, pp. 148–155. ACL.

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.

Pieraccini, R., Levin, E., and Lee, C.-H. (1991). Stochastic representation of conceptual structure in the ATIS task. In *Proceedings DARPA Speech and Natural Language Workshop*, Pacific Grove, CA, pp. 121–124. Morgan Kaufmann.

Pierrehumbert, J. and Hirschberg, J. (1990). The meaning of intonational contours in the interpretation of discourse. In Cohen, P. R., Morgan, J., and Pollack, M. (Eds.), *Intentions in Communication*, pp. 271–311. MIT Press, Cambridge, MA.

Pierrehumbert, J. (1980). *The Phonology and Phonetics of English Intonation*. Ph.D. thesis, MIT.

Polifroni, J., Hirschman, L., Seneff, S., and Zue, V. (1992). Experiments in evaluating interactive spoken language systems. In *Proceedings DARPA Speech and Natural Language Workshop*, Harriman, New York, pp. 28–33. Morgan Kaufmann.

Power, R. (1979). The organization of purposeful dialogs. *Linguistics*, *17*, 105–152.

Reichman, R. (1985). *Getting Computers to Talk Like You and Me*. MIT Press, Cambridge, MA.

Reithinger, N., Engel, R., Kipp, M., and Klesen, M. (1996). Predicting dialogue acts for a speech-to-speech translation system. In *ICSLP-96*, Philadelphia, PA, Vol. 2, pp. 654–657.

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

Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ. Second edition.

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.

Sadek, D. and De Mori, R. (1998). Dialogue systems. In De Mori, R. (Ed.), *using Parsed Corpora*. Academic Press, London.

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

Samuel, K., Carberry, S., and Vijay-Shanker, K. (1998). Dialogue act tagging with transformation-based learning. In *COLING/ACL-98*, Montreal, Vol. 2, pp. 1150–1156. ACL.

San-Segundo, R., Montero, J., Ferreiros, J., Còrdoba, R., and Pardo, J. (2001). Designing confirmation mechanisms and error recovery techniques in a railway information system for Spanish. In *In Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue, Aalborg, Denmark*.

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

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.

Searle, J. R. (1975a). Indirect speech acts. In Cole, P. and Morgan, J. L. (Eds.), *Speech Acts: Syntax and Semantics Volume 3*, pp. 59–82. Academic Press, New York.

Searle, J. R. (1975b). A taxonomy of illocutionary acts. In Gunderson, K. (Ed.), *Language, Mind and Knowledge, Minnesota Studies in the Philosophy of Science*, Vol. VII, pp. 344–369. University of Minnesota Press, Amsterdam. Also appears in John R. Searle, *Expression and Meaning: Studies in the Theory of Speech Acts*, Cambridge University Press, 1979.

Seneff, S. (1995). TINA: A natural language system for spoken language application. *Computational Linguistics*, *18*(1), 62–86.

Seneff, S. (2002). Response planning and generation in the MERCURY flight reservation system. *Computer Speech and Language, Special Issue on Spoken Language Generation*, *16*(3-4), 283–312.

Seneff, S. and Polifroni, J. (2000). Dialogue management in the mercury flight reservation system. In *ANLP/NAACL Workshop on Conversational Systems*, Seattle. Association for Computational Linguistics.

Shriberg, E., Bates, R., Taylor, P., Stolcke, A., Jurafsky, D., Ries, K., Coccaro, N., Martin, R., Meteer, M., and Ess-Dykema, C. V. (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.

Shriberg, E., Wade, E., and Price, P. (1992). Human-machine problem solving using spoken language systems (SLS): Factors affecting performance and user satisfaction. In *Proceedings DARPA Speech and Natural Language Workshop*, Harriman, New York, pp. 49–54. Morgan Kaufmann.

Singh, S. P., Litman, D. J., Kearns, M. J., and Walker, M. A. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system.. *J. Artif. Intell. Res. (JAIR)*, *16*, 105–133.

Smith, R. W. and Gordon, S. A. (1997). Effects of variable initiative on linguistic behavior in human-computer spoken natural language dialogue. *Computational Linguistics*, *23*(1), 141–168.

Smith, V. L. and Clark, H. H. (1993). On the course of answering questions. *Journal of Memory and Language*, *32*, 25–38.

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

Stent, A. (2002). A conversation acts model for generating spoken dialogue contributions. *Computer Speech and Language, Special Issue on Spoken Language Generation*, *16*(3-4).

Stifelman, L. J., Arons, B., Schmandt, C., and Hulteen, E. A. (1993). VoiceNotes: A speech interface for a hand-held voice notetaker. In *Human Factors in Computing Systems: INTERCHI '93 Conference Proceedings*, Amsterdam, pp. 179–186. ACM.

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

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

Stolcke, A., Shriberg, E., Bates, R., Coccaro, N., Jurafsky, D., Martin, R., Meteer, M., Ries, K., Taylor, P., and Van Ess-Dykema, C. (1998). Dialog act modeling for conversational speech. 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, Stanford, CA, pp. 98–105. AAAI Press.

Stolcke, A. and Shriberg, E. (1996). Automatic linguistic segmentation of conversational speech. In *ICSLP-96*, Philadelphia, PA, pp. 1005–1008.

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

Taylor, P., King, S., Isard, S., and Wright, H. (1998). Intonation and dialog context as constraints for speech recognition. *Language and Speech*, *41*(3-4), 489–508.

Traum, D. (2000). 20 questions for dialogue act taxonomies. *Journal of Semantics*, *17*(1).

Traum, D. and Larsson, S. (2003). The information state approach to dialogue management. In van Kuppevelt, J. and Smith, R. (Eds.), *Current and New Directions in Discourse and Dialogue*. Kluwer.

Traum, D. R. and Hinkelman, E. A. (1992). Conversation acts in task-oriented spoken dialogue. *Computational Intelligence: Special Issue on Computational Approaches to Non-Literal Language*, *8*(3), 575–599.

Waibel, A. (1988). *Prosody and Speech Recognition*. Morgan Kaufmann, San Mateo, CA.

Walker, M., Kamm, C., and Litman, D. (2001). Towards developing general models of usability with PARADISE. *Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems*, *6*(3).

Walker, M., Passonneau, R., Rudnicky, A., Aberdeen, J., Boland, J., Bratt, E., Garofolo, J., Hirschman, L., , Le, A., Lee, S., Narayanan, S., Papineni, K., Pellom, B., Polifroni, J., Potamianos, A., Prabhu, P., Rudnicky, A., Sanders, G., Seneff, S., Stallard, D., and Whittaker, S. (2002). Cross-site evaluation in DARPA Communicator: The June 2000 data collection. submitted.

Walker, M. A., Fromer, J. C., and Narayanan, S. (1998). Learning optimal dialogue strategies: a case study of a spoken dialogue agent for email. In *Proceedings of the 17th international conference on Computational linguistics*, Montreal, Quebec, Canada, pp. 1345–1351. ACL.

Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. (1997). PARADISE: A framework for evaluating spoken dialogue agents. In *ACL/EACL-97*, Madrid, Spain, pp. 271–280. ACL.

Walker, M. A., Maier, E., Allen, J., Carletta, J., Condon, S., Flammia, G., Hirschberg, J., Isard, S., Ishizaki, M., Levin, L., Luperfoy, S., Traum, D., and Whittaker, S. (1996). Penn multiparty standard coding scheme: Draft annotation manual. `www.cis.upenn.edu/˜ircs/discourse-tagging/newcoding.html`.

Walker, M. A. and Rambow, O. C. (2002). Spoken language generation. *Computer Speech and Language, Special Issue on Spoken Language Generation*, *16*(3-4), 273–281.

Walker, M. A. and Whittaker, S. (1990). Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings of the 28th ACL*, Pittsburgh, PA, pp. 70–78. ACL.

Walker, M. *et al*. (2001). Cross-site evaluation in darpa communicator: The june 2000 data collection. Submitted ms.

Ward, W. and Issar, S. (1994). Recent improvements in the cmu spoken language understanding system. In *ARPA Human Language Technologies Workshop*, Plainsboro, N.J.

Warnke, V., Kompe, R., Niemann, H., and Nöth, E. (1997). Integrated dialog act segmentation and classification using prosodic features and language models. In *EUROSPEECH-97*, Vol. 1, pp. 207–210.

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

Williams, J. D. and Young, S. (2003). Using wizard-of-oz simulations to bootstrap reinforcement-learning-based dialog management systems. In *Proceedings of the 4th SIGDIAL Workshop on Discourse and Dialogue*, Sapporo, Japan. Association for Computational Linguistics.

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

Woszczyna, M. and Waibel, A. (1994). Inferring linguistic structure in spoken language. In *ICSLP-94*, Yokohama, Japan, pp. 847–850.

Xu, W. and Rudnicky, A. I. (2000). Task-based dialog management using an agenda. In *ANLP/NAACL Workshop on Conversational Systems*, Somerset, New Jersey, pp. 42–47. Association for Computational Linguistics.

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. ACM.

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

Young, S. (2002). The statistical approach to the design of spoken dialogue systems. Tech. rep. CUED/F-INFENG/TR.433, Cambridge University Engineering Department, Cambridge, England.

Zue, V., Glass, J., Goodine, D., Leung, H., Phillips, M., Polifroni, J., and Seneff, S. (1989). Preliminary evaluation of the VOYAGER spoken language system. In *Proceedings DARPA Speech and Natural Language Workshop*, Cape Cod, MA, pp. 160–167. Morgan Kaufmann.