CHAPTER

26 Dialogue Systems and Chatbots

Les lois de la conversation sont en général de ne s'y appesantir sur aucun objet, mais de passer légèrement, sans effort et sans affectation, d'un sujet à un autre ; de savoir y parler de choses frivoles comme de choses sérieuses

The rules of conversation are, in general, not to dwell on any one subject, but to pass lightly from one to another without effort and without affectation; to know how to speak about trivial topics as well as serious ones;

The 18th C. *Encyclopedia* of Diderot, start of the entry on conversation

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, there is something deeply moving about creating something and then having a chat with it. Legend has it that after finishing his sculpture *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 is the mark of humanity and sentience, and **conversation** or **dialogue** is the most fundamental and specially privileged arena of language. It is 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 spinach, participating in business meetings or talking with our families, booking airline flights or complaining about the weather.

conversation
dialogue

This chapter introduces the fundamental algorithms of **conversational agents**, or **dialogue systems**. These programs communicate with users in natural language (text, speech, or both), and fall into two classes. **Task-oriented dialogue agents** use conversation with users to help complete tasks. Dialogue agents in digital assistants (Siri, Alexa, Google Now/Home, Cortana, etc.), give directions, control appliances, find restaurants, or make calls. Conversational agents can answer questions on corporate websites, interface with robots, and even be used for social good: DoNotPay is a "robot lawyer" that helps people challenge incorrect parking fines, apply for emergency housing, or claim asylum if they are refugees. By contrast, **chatbots** are systems designed for extended conversations, set up to mimic the unstructured conversations or 'chats' characteristic of human-human interaction, mainly for entertainment, but also for practical purposes like making task-oriented agents more natural. In Section 26.2 we'll discuss the three major chatbot architectures : rule-based systems, information retrieval systems, and encoder-decoder models. In Section 26.3 we turn to task-oriented agents, introducing the frame-based architecture (the **GUS** architecture) that underlies most modern task-based systems.

conversational agent
dialogue system

26.1 Properties of Human Conversation

Conversation between humans is an intricate and complex joint activity. Before we attempt to design a conversational agent to converse with humans, it is crucial to understand something about how humans converse with each other. Consider some of the phenomena that occur in the conversation between a human travel agent and a human client excerpted in Fig. 26.1.

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.
A_7:	And you said returning on May 15th?
C_8:	Uh, yeah, at the end of the day.
A_8:	OK. There's #two non-stops . . . #
C_9:	#Act. . . actually #, what day of the week is the 15th?
A_9:	It's a Friday.
C_{10}:	Uh hmm. I would consider staying there an extra day til Sunday.
A_{10}:	OK. . . OK. On Sunday I have . . .

Figure 26.1 Part of a phone conversation between a human travel agent (A) and human client (C). The passages framed by # in A_8 and C_9 indicate overlaps in speech.

Turns

turn

A dialogue is a sequence of **turns** (A_1, B_1, A_2, and so on), each a single contribution to the dialogue (as if in a game: I take a turn, then you take a turn, then me, and so on). A turn can consist of a sentence (like C_1), although it might be as short as a single word (C_7) or as long as multiple sentences (A_5).

Turn structure has important implications for spoken dialogue. A system has to know when to stop talking; the client interrupts (in A_8 and C_9), so the system must know to stop talking (and that the user might be making a correction). A system also has to know when to start talking. For example, most of the time in conversation, speakers start their turns almost immediately after the other speaker finishes, without a long pause, because people are able to (most of the time) detect when the other person is about to finish talking. Spoken dialogue systems must also detect whether a user is done speaking, so they can process the utterance and respond. This task—called **endpointing** or **endpoint detection**— can be quite challenging because of noise and because people often pause in the middle of turns.

endpointing

Speech Acts

speech acts

A key insight into conversation—due originally to the philosopher Wittgenstein (1953) but worked out more fully by Austin (1962)—is that each utterance in a dialogue is a kind of **action** being performed by the speaker. These actions are commonly called **speech acts** or **dialog acts**: here's one taxonomy consisting of 4 major classes (Bach and Harnish, 1979):

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

A user asking a person or a dialogue system to do something ('Turn up the music') is issuing a DIRECTIVE. Asking a question that requires an answer is also a way of issuing a DIRECTIVE: in a sense when the system says (C_2) "what day in May did you want to travel?" it's as if the system is (very politely) commanding the system to answer. By contrast, a user stating a constraint (like C_1 'I need to travel in May') is issuing a CONSTATIVE. A user thanking the system is issuing an ACKNOWLEDGMENT. The speech act expresses an important component of the intention of the speaker (or writer) in saying what they said.

Grounding

common ground

grounding

A dialogue is not just a series of independent speech acts, but rather a collective act performed by the speaker and the hearer. Like all collective acts, it's important for the participants to establish what they both agree on, called the **common ground** (Stalnaker, 1978). Speakers do this by **grounding** each other's utterances. Grounding means acknowledging that the hearer has understood the speaker; like an ACK used to confirm receipt in data communications (Clark, 1996). (People need grounding for non-linguistic actions as well; the reason an elevator button lights up when it's pressed is to acknowledge that the elevator has indeed been called (Norman, 1988).)

Humans constantly ground each other's utterances. We can ground by explicitly saying "OK", as the agent does in A_8 or A_{10}. Or we can ground by repeating what the other person says; in utterance A_1 the agent repeats "in May", demonstrating her understanding to the client. Or notice that when the client answers a question, the agent begins the next question with "And". The "And" implies that the new question is 'in addition' to the old question, again indicating to the client that the agent has successfully understood the answer to the last question.

Subdialogues and Dialogue Structure

conversational analysis

adjacency pair

Conversations have structure. Consider, for example, the local structure between speech acts discussed in the field of **conversational analysis** (Sacks et al., 1974). QUESTIONS set up an expectation for an ANSWER. PROPOSALS are followed by ACCEPTANCE (or REJECTION). COMPLIMENTS ("Nice jacket!") often give rise to DOWNPLAYERS ("Oh, this old thing?"). These pairs, called **adjacency pairs** are composed of a **first pair part** and a **second pair part** (Schegloff, 1968), and these expectations can help systems decide what actions to take.

side sequence
subdialogue

However, dialogue acts aren't always followed immediately by their second pair part. The two parts can be separated by a **side sequence** (Jefferson 1972) or **subdialogue**. For example utterances C_9 to A_{10} constitute a **correction subdialogue** (Litman 1985, Litman and Allen 1987, Chu-Carroll and Carberry 1998):

C_9: #Act... actually#, what day of the week is the 15th?
A_9: It's a Friday.
C_{10}: Uh hmm. I would consider staying there an extra day til Sunday.
A_{10}: OK...OK. On Sunday I have ...

The question in C_9 interrupts the prior discourse, in which the agent was looking for a May 15 return flight. The agent must answer the question and also realize that ''I would consider staying...til Sunday" means that the client would probably like to change their plan, and now go back to finding return flights, but for the 17th.

Another side sequence is the **clarification question**, which can form a subdialogue between a REQUEST and a RESPONSE. This is especially common in dialogue systems where speech recognition errors causes the system to have to ask for clarifications or repetitions like the following:

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

presequence

In addition to side-sequences, questions often have **presequences**, like the following example where a user starts with a question about the system's capabilities ("Can you make train reservations") before making a request.

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

Initiative

initiative

Sometimes a conversation is completely controlled by one participant. For example a reporter interviewing a chef might ask questions, and the chef responds. We say that the reporter in this case has the conversational **initiative** (Walker and Whittaker, 1990). In normal human-human dialogue, however, it's more common for initiative to shift back and forth between the participants, as they sometimes answer questions, sometimes ask them, sometimes take the conversations in new directions, sometimes not. You may ask me a question, and then I respond asking you to clarify something you said, which leads the conversation in all sorts of ways. We call such interactions **mixed initiative**.

Mixed initiative, while the norm for human-human conversations, is very difficult for dialogue systems to achieve. It's much easier to design dialogue systems to be passive responders. In the question answering systems we saw in Chapter 25, or in simple search engines, the initiative lies completely with the user. In such **user-initiative** systems, the user specifies a query, and the systems responds. Then the user can specify another query. Alternatively, you may have had the experience of being stuck in a bad dialogue system that asks a question and gives you no opportunity to do anything until you answer it. Such **system-initiative** architectures can be very frustrating.

Inference and Implicature

Inference is also important in dialogue understanding. Consider the client's response C_2, 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 agent's question. The client merely mentions a meeting at a certain time. 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?

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. This kind of example was pointed out by Grice (1975, 1978) as part of his theory of **conversational implicature**. **Implicature** means a particular class of licensed inferences. Grice proposed that what enables hearers to draw these inferences is that conversation is guided by a set of **maxims**, general heuristics that play a guiding role in the interpretation of conversational utterances. One such maxim is the maxim of **relevance** which says that speakers attempt to be relevant, they don't just utter random speech acts. When the client mentions a meeting on the 12th, the agent reasons 'There must be some relevance for mentioning this meeting. What could it be?'. The agent knows that one precondition for having a meeting (at least before Web conferencing) is being at the place where the meeting is held, and therefore that maybe the meeting is a reason for the travel, and if so, then since people like to arrive the day before a meeting, the agent should infer that the flight should be on the 11th.

implicature

relevance

These subtle characteristics of human conversations (**turns**, **speech acts**, **grounding**, **dialogue structure**, **initiative**, and **implicature**) are among the reasons it is difficult to build dialogue systems that can carry on natural conversations with humans. Many of these challenges are active areas of dialogue systems research.

26.2 Chatbots

chatbot

The simplest kinds of dialogue systems are **chatbots**, systems that can carry on extended conversations with the goal of mimicking the unstructured conversations or 'chats' characteristic of informal human-human interaction.

These systems often have an entertainment value, such as Cleverbot, a simple IR-based chatbot capable of carrying on the kinds of conversations shown in Fig. 26.2, or Microsoft's XiaoIce system[1] (Zhou et al., 2018), a more sophisticated commercial system that chats with people on text messaging platforms (Fig. 26.3). Yet starting from the very first system, ELIZA (Weizenbaum, 1966), chatbots have also been used for practical purposes, such as testing theories of psychological counseling.

Like practically everything else in language processing, chatbot architectures fall into two classes: rule-based systems and corpus-based systems. Rule-based systems include the early influential ELIZA and PARRY systems. Corpus-based systems mine large datasets of human-human conversations, which can be done by using information retrieval (IR-based systems simply copy a human's response from

[1] XiaoIce is a somewhat confusing translation of the systems Chinese name 小冰, 'Xiao Bing', which means Little Bing or Little Ice.

User:	Do you like Doctor Who?
Cleverbot:	Yes, so funny.
User:	What's your favorite show?
Cleverbot:	I don't have one.
User:	What's your name?
Cleverbot:	Jason I already told you.
User:	No your real name.
Cleverbot:	Stephanie IS my real name.
User:	I thought it was Jason?
Cleverbot:	How did you come to that conclusion.
User:	Because you said so.
Cleverbot:	I have to do my homework.

Figure 26.2 Chatbots: A conversation between a user and Cleverbot, an IR-based chatbot (Carpenter, 2017).

[人类]	小冰你好	[Human]	Hello, XiaoIce.
[小冰]	你好呀，你头像好有趣，感觉在哪见过	[XiaoIce]	Hello, your profile pic is interesting. Like I've seen it somewhere.
[人类]	是吗	[Human]	Really?
[小冰]	提不是哪部动画里的呀	[XiaoIce]	Is it a comic character?
[人类]	你怎么知道的	[Human]	How do you know that?
[小冰]	你猜	[XiaoIce]	Guess.
[人类]	难道你是真人吗?	[Human]	Are you human?
[小冰]	这个，你和我多聊聊就知道了	[XiaoIce]	Well, you'll know it after chatting with me more.
[人类]	你不是人工智能机器人么	[Human]	Aren't you an AI chatbot?
[小冰]	是又怎样，我不会冷冰冰的对待你的	[XiaoIce]	So what? I won't be indifferent to you.
[人类]	你都有什么功能啊	[Human]	What functions do you have?
[小冰]	我有200多个skills，来，说出你的需求	[XiaoIce]	I have over 200 skills. Just name one you need.

Figure 26.3 A sample Chinese conversation from the XiaoIce chatbot, together with translation from Zhou et al. (2018).

a previous conversation) or by using a machine translation paradigm such as neural network sequence-to-sequence systems, to learn to map from a user utterance to a system response.

26.2.1 Rule-based chatbots: ELIZA and PARRY

ELIZA (Weizenbaum, 1966) is the most important chatbot dialogue system in the history of the field, and so we take some time here to describe its algorithm in detail. ELIZA was designed to simulate a Rogerian psychologist, based on a branch of clinical psychology whose methods involve drawing the patient out by reflecting patient's statements back at them. Rogerian psychology is the rare type of conversation in which, as Weizenbaum points out, one can "assume the pose of knowing almost nothing of the real world". If a patient says "I went for a long boat ride" and the psychiatrist says "Tell me about boats", you don't assume she didn't know what a boat is, but rather assume she had some conversational goal. Most chatbots trying to pass the Turing test choose a domain with similar properties.

Fig. 26.4 shows the most famous ELIZA conversation, in which ELIZA seems

Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED
..
..
WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?
My father
YOUR FATHER
You are like my father in some ways.
WHAT RESEMBLANCE DO YOU SEE
You are not very aggressive but I think you don't want me to notice that.
WHAT MAKES YOU THINK I AM NOT AGGRESSIVE
You don't argue with me.
WHY DO YOU THINK I DON'T ARGUE WITH YOU
You are afraid of me.
DOES IT PLEASE YOU TO BELIEVE I'M AFRAID OF YOU
My father is afraid of everybody.
WHAT ELSE COMES TO MIND WHEN YOU THINK OF YOUR FATHER
Bullies.
DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE

Figure 26.4 Sample ELIZA dialogue from Weizenbaum (1966).

to simulate giving very sophisticated therapeutic advice, excerpted (deleting some material with ...) from a full interaction given in the original paper (Weizenbaum, 1966).

As we summarized briefly in Chapter 2, ELIZA worked by pattern/transform rules like the following one:

```
(0 YOU 0 ME) [pattern]
->
(WHAT MAKES YOU THINK I 3 YOU) [transform]
```

In the ELIZA pattern, 0 means Kleene*, and in the transform rules the numbers are the index of the constituent in the pattern. Thus the number 3 refers to the second 0 in the first pattern. This rule would transfer

```
You hate me
```

into:

```
WHAT MAKES YOU THINK I HATE YOU
```

Each ELIZA pattern/rule is linked to a keyword that might occur in a user sentence. The architecture is summarized in Fig. 26.5.

Keywords are associated with a rank, with specific words being more highly ranked, and more general words ranking lower. Consider the following user sentence:

```
function ELIZA GENERATOR(user sentence) returns response

Find the word w in sentence that has the highest keyword rank
  if w exists
     Choose the highest ranked rule r for w that matches sentence
     response ← Apply the transform in r to sentence
     if w = 'my'
        future ← Apply a transformation from the 'memory' rule list to sentence
        Push future onto memory stack
   else (no keyword applies)
    either
      response ← Apply the transform for the NONE keyword to sentence
    or
      response ← Pop the top response from the memory stack
 return(response)
```

Figure 26.5 A simplified sketch of the ELIZA algorithm. The power of the algorithm comes from the particular transforms associated with each keyword.

```
I know everybody laughed at me
```

Because it has the word "I", this sentence could match the following rule whose keyword is *I*:

```
(I *) -> (You say you 2)
```

producing:

```
YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU
```

But as Weizenbaum points out, "I" is a very general word and its keywords lead to very general responses. Instead the keyword "everybody" is much more interesting, since someone using universals like everybody or always is probably "referring to some quite specific event or person". Therefore, ELIZA prefers to respond with the pattern associated with the more specific keyword *everybody* (implementing by just assigning "everybody" rank 5 and "I" rank 0 in the lexicon), whose rule thus transforms the sentence to:

```
WHO IN PARTICULAR ARE YOU THINKING OF?
```

If no keyword matches, ELIZA chooses a non-commital response like "PLEASE GO ON", "THAT'S VERY INTERESTING", or "I SEE".

Finally, ELIZA has a clever memory trick that accounts for the very last sentence of the conversation above. Whenever the word "my" is the highest ranked keyword, ELIZA will randomly select a transform on the MEMORY list, apply it to the sentence, and store it on the stack:

```
(MEMORY MY
  (0 MY 0 = LETS DISCUSS FURTHER WHY YOUR 3)
  (0 MY 0 = EARLIER YOU SAID YOUR 3)
  (0 MY 0 = DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR 3
```

Later, if no keyword matches a sentence, ELIZA will return the top of the MEMORY queue instead. [2]

[2] Fun fact: because of its structure as a queue, this MEMORY trick is the earliest known hierarchical model of discourse in natural language processing.

People became deeply emotionally involved with the program. Weizenbaum tells the story of one of his staff who would ask Weizenbaum to leave the room when she talked with ELIZA. When Weizenbaum suggested that he might want to store all the ELIZA conversations for later analysis, people immediately pointed out the privacy implications, which suggested that they were having quite private conversations with ELIZA, despite knowing that it was just software.

ELIZA's framework is still used today; modern chatbot system tools like ALICE are based on updated versions of ELIZA's pattern/action architecture.

A few years after ELIZA, another chatbot with a clinical psychology focus, PARRY (Colby et al., 1971), was used to study schizophrenia. In addition to ELIZA-like regular expressions, the PARRY system included a model of its own mental state, with affect variables for the agent's levels of fear and anger; certain topics of conversation might lead PARRY to become more angry or mistrustful. 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. Parry was the first known system to pass the Turing test (in 1972!); psychiatrists couldn't distinguish text transcripts of interviews with PARRY from transcripts of interviews with real paranoids (Colby et al., 1972).

26.2.2 Corpus-based chatbots

Corpus-based chatbots, instead of using hand-built rules, mine conversations of human-human conversations, (or sometimes mine the human sides of human-machine conversations).

These systems are enormously data-intensive; Serban et al. (2018) estimate that training modern chatbots require hundreds of millions or even billions of words. Many such corpora have been used, including large spoken conversational corpora like the Switchboard corpus of American English telephone conversations (Godfrey et al., 1992) or the various CALLHOME and CALLFRIEND telephone conversational corpora in many languages. Many systems also train on movie dialogue, which is available in great quantities in various corpora (Lison and Tiedemann, 2016, inter alia), and which resembles natural conversation in many ways (Forchini, 2013). Text from microblogging sites like Twitter (Ritter et al., 2010) or a Weibo (微博) have also been used, or datasets of crowdworker conversations like Topical-Chat (Gopalakrishnan et al., 2019). Many corpora also focus on specific topics, and can be used for topical chatbots. See Serban et al. (2018) for a comprehensive summary of available corpora. Another common technique is to extract possible responses from non-dialogue corpora, so that a chatbot can tell stories or mention facts acquired in that way.

Finally, once a chatbot has been put into practice, the turns that humans use to respond to the chatbot can be used as additional conversational data for training. The XiaoIce system collects and stores all human-machine conversations between XiaoIce and its users, resulting in a a dataset of over 30 billion conversation pairs. It's crucial in these cases to remove personally identifiable information (PII); see Section 26.6.1.

The two main architectures for corpus-based chatbots: information retrieval, and machine learned sequence transduction. Like rule-based chatbots (but unlike frame-based dialogue systems), most corpus-based chatbots do very little modeling of the conversational context. Instead they tend to focus on generating a single response turn that is appropriate given the user's immediately previous utterance or two. For response generation

this reason they are often called **response generation** systems. Corpus-based chatbots thus have some similarity to question answering systems, which focus on single responses while ignoring context or larger conversational goals.

IR-based chatbots

The principle behind information retrieval based chatbots is to respond to a user's turn X by repeating some appropriate turn Y from a corpus of natural (human) text of the sort described in the prior section.

Given the corpus and the user's sentence, IR-based systems can use any retrieval algorithm to choose an appropriate response from the corpus. The two simplest methods are the following:

1. Return the response to the most similar turn: Given user query q and a conversational corpus C, find the turn t in C that is most similar to q (for example has the highest cosine with q) and return the *following* turn, i.e. the human response to t in C:

$$r = response\left(\underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{||q||t||}\right) \tag{26.1}$$

The idea is that we should look for a turn that most resembles the user's turn, and return the human response to that turn (Jafarpour et al. 2009, Leuski and Traum 2011).

2. Return the most similar turn: Given user query q and a conversational corpus C, return the turn t in C that is most similar to q (for example has the highest cosine with q):

$$r = \underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{||q||t||} \tag{26.2}$$

The idea here is to directly match the users query q with turns from C, since a good response will often share words or semantics with the prior turn.

In each case, any similarity function can be used, such as cosines computed either over words (weighted by tf-idf) or more commonly now, cosines over any kind of sentence embeddings.

Although returning the *response* to the most similar turn seems like a more intuitive algorithm, returning the most similar turn seems to work better in practice, perhaps because selecting the response adds another layer of indirection that can allow for more noise (Ritter et al. 2011, Wang et al. 2013).

The IR-based approach can be extended by using more features than just the words in the q. For example using the entire conversation with the user so far can be quite helpful when the user's query is short (like "Yes" or "OK"). Information about the user or sentiment or other information can also play a role. The IR-based approach can even draw responses from narrative (non-dialogue) text. The COBOT chatbot (Isbell et al., 2000) pioneered this approach, generating responses by selecting sentences from a corpus that combined the Unabomber Manifesto by Theodore Kaczynski, articles on alien abduction, the scripts of "The Big Lebowski" and "Planet of the Apes". Chatbots that want to generate informative turns such as answers to user questions can use texts like Wikipedia to draw on sentences that might contain those answers (Yan et al., 2016). XiaoIce similarly collects sentences from public lectures and news articles and searches them using IR based on query expansion from the user's turn to respond to turns like "Tell me something about Beijing" (Zhou et al., 2018).

Encoder decoder chatbots

"What ho!" I said.
"What ho!" said Motty.
"What ho! What ho!"
"What ho! What ho! What ho!"
After that it seemed rather difficult to go on with the conversation."
Wodehouse *My Man Jeeves*

An alternate way to use a corpus to generate dialogue is to think of response generation as a task of *transducing* from the user's prior turn to the system's turn. This is basically the machine learning version of Eliza; the system learns from a corpus to transduce a question to an answer.

This idea was first developed by using phrase-based machine translation (Ritter et al., 2011) to translate a user turn to a system response. It quickly became clear, however, that the task of response generation was too different from machine translation. In machine translation, words or phrases in the source and target sentences tend to align well with each other, but in conversation, a user utterance may share no words or phrases with a coherent response.

Instead, (roughly contemporaneously by Shang et al. 2015, Vinyals and Le 2015, and Sordoni et al. 2015) transduction models for response generation were modeled instead using encoder-decoder models (Chapter 11), as shown in Fig. 26.6.

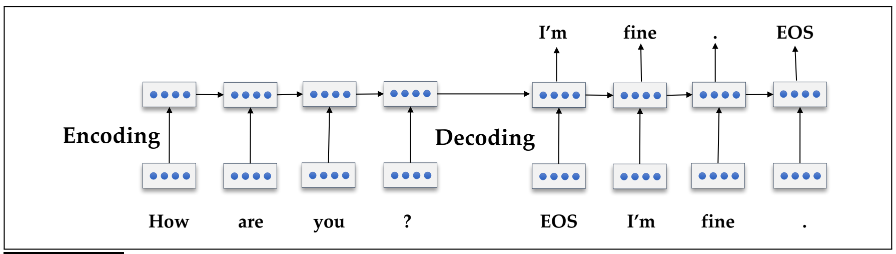

Figure 26.6 An encoder decoder model for neural response generation in dialogue.

A number of modifications are required to the basic encoder-decoder model to adapt it for the task of response generation. For example basic encoder-decoder models have a tendency to produce predictable but repetitive and therefore dull responses like "I'm OK" or "I don't know" that shut down the conversation. This can be addressed by changing the objective function for seq2seq model training to a mutual information objective, or by modifying a beam decoder to keep more diverse responses in the beam (Li et al., 2016a).

Another problem with simple encoder-decoder response generation architecture is its inability to model the longer prior context of the conversation. This can be done by allowing the model to see prior turns, such as by using a hierarchical model that summarizes information over multiple prior turns (Lowe et al., 2017b).

Finally, encoder-decoder response generators focus on generating single responses, and so don't tend to do a good job of continuously generating responses that cohere across multiple turns. This can be addressed by using reinforcement learning, as well as techniques like adversarial networks, to learn to choose responses that make the overall conversation more natural (Li et al. 2016b, Li et al. 2017).

Fig. 26.7 shows some sample responses generated by a vanilla SEQ2SEQ model, and from a model trained by an adversarial algorithm to produce responses that are harder to distinguish from human responses (Li et al., 2017).

Input	tell me ... how long have you had this falling sickness ?
Vanilla-SEQ2SEQ	i'm not a doctor.
Adversarial	a few months, i guess .
Input	so i had the doctors test sammy 's response to conditioning .
Vanilla-SEQ2SEQ	sammy wrote the test sammy wrote the test .
Adversarial	so he took the pills .
Input	they didn 't have much success with this problem commander .
Vanilla-SEQ2SEQ	they 're not the only ones who have been in the system .
Adversarial	can we find someone else ?

Figure 26.7 Sample responses generated by a SEQ2SEQ model trained either with a vanilla maximum likelihood objective, or adversarially trained to produce sentences that are hard for an adversary to distinguish from human sentences (Li et al., 2017).

Evaluating Chatbots

Chatbots are generally evaluated by humans. The slot-filling evaluations used for task-based dialogue (Section 26.5) aren't appropriate for this task (Artstein et al., 2009), and word-overlap metrics like BLEU for comparing a chatbot's response to a human response turn out to correlate very poorly with human judgments (Liu et al., 2016). BLEU performs poorly because there are so many possible responses to any given turn; word-overlap metrics work best when the space of responses is small and lexically overlapping, as is the case in machine translation.

While human evaluation is therefore required for evaluating chatbots, there are beginning to be models for automatic evaluation. The ADEM (Lowe et al., 2017a) classifier is trained on a set of responses labeled by humans with how appropriate they are, and learns to predict this label from the dialogue context and the words in the system response.

adversarial evaluation

Another paradigm is **adversarial evaluation** (Bowman et al. 2016, Kannan and Vinyals 2016, Li et al. 2017), inspired by the Turing test. The idea is to train a "Turing-like" evaluator classifier to distinguish between human-generated responses and machine-generated responses. The more successful a response generation system is at fooling this evaluator, the better the system.

26.3 GUS: Simple Frame-based Dialogue Systems

GUS

We turn now to **task-based dialogue**, in which a dialogue system has the goal of helping a user solve some task like making an airplane reservation or buying a product. In this section we introduce the very simple **GUS** architecture for task-based dialogue. Although it is quite old, (it was first introduced in 1977 in the influential GUS system for travel planning (Bobrow et al., 1977)) the GUS architecture has been astonishingly long-lived and underlies most if not all modern commercial digital assistants. Figure 26.8 shows a conversation with the original GUS system.

dialogue state

frame

slot

All modern task-based dialogue systems, whether the simple GUS architecture we describe here, or the more sophisticated **dialogue state** architectures we turn to in the following section, are based around **frames**. A frame is a kind of knowledge structure representing the kinds of intentions the system can extract from user sentences, and consists of a collection of **slots**, each of which can take a set of possible **values**. Together this set of frames is sometimes called a **domain ontology**.

The set of slots in a task-based dialogue frame specifies what the system needs to know, and the filler of each slot is constrained to values of a particular semantic

GUS:	Hello. My name is GUS. I can help you plan a simple trip by air. Where do you want to go ?
Client:	I want to go to San Diego on May 28
GUS:	What time do you want to leave?
Client:	I must be in San Diego before 10 am
GUS:	Would you like P.S.A. flight 102 that arrives at 9.15 am ?
Client:	What is the next flight ?
GUS:	Air California flight 310 that leaves at 8.30am and arrives at 10.00am
Client:	I'll take the first one
GUS:	In what name should I make a reservation, please?
Client:	Dan Bobrow
GUS:	I have confirmed the following flight: P.S.A. flight 102 on Wednesday May 28 from San Jose to San Diego that leaves at 7.30 am and arrives at 9.15 am.
GUS:	What date do you want to return on ?
Client:	On Friday in the evening.
GUS:	Would you like the flight that leaves at 7.45 pm ?
Client:	That's fine.
GUS:	I have confirmed the following flight: P.S.A. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm Thank you for calling. Goodbye

Figure 26.8 The travel domain: A transcript of an actual dialogue with the GUS system of Bobrow et al. (1977). P.S.A. and Air California were airlines of that period.

type. In the travel domain, for example, a slot might be of type city (hence take on values like *San Francisco*, or *Hong Kong*) or of type date, airline, or time.

Slot	**Type**	Question Template
ORIGIN CITY	city	"From what city are you leaving?"
DESTINATION CITY	city	"Where are you going?"
DEPARTURE TIME	time	"When would you like to leave?"
DEPARTURE DATE	date	"What day would you like to leave?"
ARRIVAL TIME	time	"When do you want to arrive?"
ARRIVAL DATE	date	"What day would you like to arrive?"

Figure 26.9 A frame in a frame-based dialogue system, showing the type of each slot and a question used to fill the slot.

Types in GUS, as in modern frame-based dialogue agents, have hierarchical structure; for example the *date* type in GUS is itself a frame with slots with types like *integer* or members of sets of weekday names:

```
DATE
  MONTH:NAME   YEAR:INTEGER    DAY:(BOUNDED-INTEGER 1 31)
  WEEKDAY:(MEMBER (Sunday Monday Tuesday Wednesday
                   Thursday Friday Saturday))
```

26.3.1 Control structure for frame-based dialogue

The control architecture for frame-based dialogue systems, used in various forms in modern systems like Apple's Siri, Amazon's Alexa, and the Google Assistant, is designed around the frame. The system's goal is to fill the slots in the frame with the fillers the user intends, and then perform the relevant action for the user (answering a question, or booking a flight).

To do this, the system asks questions of the user (using pre-specified question templates associated with each slot of each frame, as shown in Fig. 26.9), filling any slot that the user specifies (we'll describe how slot-filling works in the next section). If a user's response fills multiple slots, like the following:

(26.3) I want a flight from San Francisco to Denver one way leaving after five p.m. on Tuesday.

the system fills all the relevant slots, and then continues asking questions to fill the remaining slots, skipping questions associated with filled slots. The GUS architecture also has condition-action rules attached to slots. For example, a rule attached to the DESTINATION slot for the plane booking frame, once the user has specified the destination, might automatically enter that city as the default *StayLocation* for the related hotel booking frame. Or if the user specifies the DESTINATION DAY for a short trip the system could automatically enter the ARRIVAL DAY.

Many domains require multiple frames. Besides frames for car or hotel reservations, we might need frames with general route information (for questions like *Which airlines fly from Boston to San Francisco?*), or information about airfare practices (for questions like *Do I have to stay a specific number of days to get a decent airfare?*). 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, the GUS architecture is a **production rule** system. 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 according to factors such as the user's input and some simple dialogue history like the last question that the system asked.

Once the system has enough information it performs the necessary action (like querying a database of flights) and returns the result to the user.

26.3.2 Natural language understanding for filling slots in GUS

The goal of the natural language understanding component in the frame-based architecture is to extract three things from the user's utterance. The first task is **domain classification**: is this user for example talking about airlines, programming an alarm clock, or dealing with their calendar? Of course this 1-of-n classification tasks is unnecessary for single-domain systems that are focused on, say, only calendar management, but multi-domain dialogue systems are the modern standard. The second is user **intent determination**: what general task or goal is the user trying to accomplish? For example the task could be to Find a Movie, or Show a Flight, or Remove a Calendar Appointment. Finally, we need to do **slot filling**: extract the particular slots and fillers that the user intends the system to understand from their utterance with respect to their intent. From a user utterance like this one:

intent determination

slot filling

```
Show me morning flights from Boston to San Francisco on Tuesday
```

a system might want to build a representation like:

```
DOMAIN:       AIR-TRAVEL
INTENT:       SHOW-FLIGHTS
ORIGIN-CITY:  Boston
ORIGIN-DATE:  Tuesday
ORIGIN-TIME:  morning
DEST-CITY:    San Francisco
```

while an utterance like

```
Wake me tomorrow at 6
```

should give an intent like this:

```
DOMAIN:  ALARM-CLOCK
INTENT:  SET-ALARM
TIME:    2017-07-01 0600-0800
```

The slot-filling method used in the original GUS system, and still quite common in industrial applications, is to use handwritten rules, often as part of the condition-action rules attached to slots or concepts. For example we might just define a regular expression for recognizing the SET-ALARM intent:

```
wake me (up) | set (the|an) alarm | get me up
```

semantic grammar

Rule-based research systems like the Phoenix system (Ward and Issar, 1994) consist of large hand-designed **semantic grammars** with thousands of rules. A semantic grammar is a context-free grammar in which the left-hand side of each rule corresponds to the semantic entities being expressed (i.e., the slot names) as in the following fragment:

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

Semantic grammars can be parsed by any CFG parsing algorithm (see Chapter 13), resulting in a hierarchical labeling of the input string with semantic node labels, as shown in Fig. 26.10.

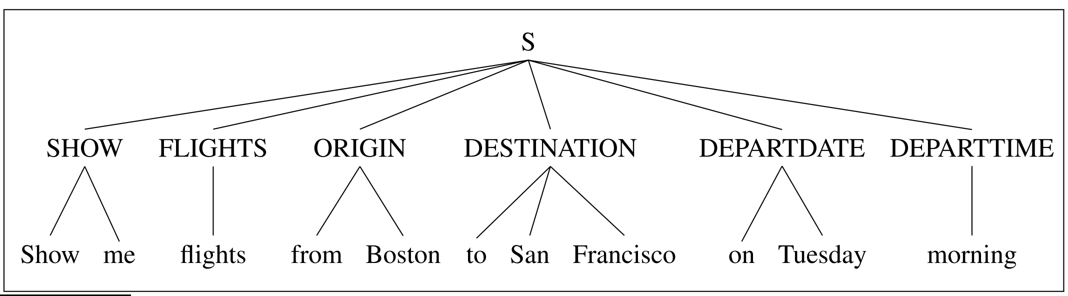

Figure 26.10 A semantic grammar parse for a user sentence, using slot names as the internal parse tree nodes.

It remains only to put the fillers into some sort of canonical form, for example by normalizing dates as discussed in Chapter 18.

Many industrial dialogue systems employ the GUS architecture but use supervised machine learning for slot-filling instead of these kinds of rules; see Section 26.4.2.

26.3.3 Other components of frame-based dialogue

The ASR (automatic speech recognition) component takes audio input from a phone or other device and outputs a transcribed string of words, as discussed in Chapter 28. The ASR component can also be made dependent on the dialogue state. For example, if the system has just asked the user "What city are you departing from?", the

ASR language model component can be constrained to assign high probability to answers with city names. This can be done by training a language model on answers to this question, or by hand-writing finite-state or context-free grammars for such sentences. A language model that is completely dependent on dialogue state is called a **restrictive grammar**; usually real systems mix restrictive grammars with more general language models.

restrictive grammar

The **natural language generation** module of any dialogue system produces the utterances that the system says to the user. Frame-based systems tend to use **template-based generation**, in which all or most of the words in the sentence to be uttered to the user are prespecified by the dialogue designer. Sentences created by these templates are often called **prompts**. Templates might be completely fixed (like 'Hello, how can I help you?'), or can include some variables that are filled in by the generator, as in the following:

template-based generation

prompt

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

It's possible to do some simple grounding even in templated generation. Consider the unnaturalness of this example from Cohen et al. (2004):

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

Without an acknowledgment, the caller doesn't know that the system has understood her 'No'. The use of *Okay* below adds grounding to the templated response *What's next?*, making (26.5) a much more natural response than (26.4):

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

The rule-based GUS approach is very common in industrial applications. As was true with the rule-based approach to information extraction, it has the advantage of high precision, and if the domain is narrow enough and experts are available, can provide sufficient coverage as well. On the other hand, the handwritten rules or grammars can be both expensive and slow to create, and handwritten rules can suffer from recall problems.

26.4 The Dialogue-State Architecture

Modern research systems for task-based dialogue are based on a more sophisticated version of the frame-based architecture called the **dialogue-state** or **belief-state** architecture. Figure 26.11 shows the six components of a typical dialogue-state system. The speech recognition and synthesis components deal with spoken language processing; we'll return to them in Chapter 28.

For the rest of this chapter we therefore consider the other four components, which are part of both spoken and textual dialogue systems. These four components are more complex than in the simple GUS systems. For example, like the GUS systems, the dialogue-state architecture has an **NLU component** to extract slot fillers from the user's utterance, but generally using machine learning rather than rules. The **dialogue state tracker** maintains the current state of the dialogue (which include the user's most recent dialogue act, plus the entire set of slot-filler constraints the user

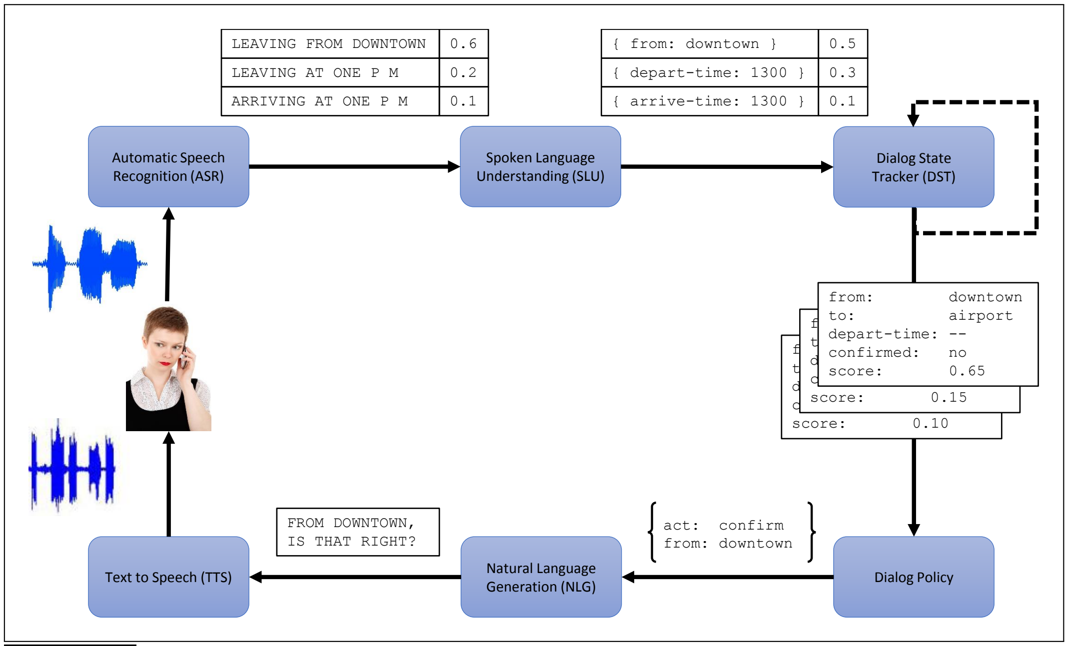

Figure 26.11 Architecture of a dialogue-state system for task-oriented dialogue from Williams et al. (2016).

has expressed so far). The **dialogue policy** decides what the system should do or say next. The dialogue policy in GUS was simple: ask questions until the frame was full and then report back the results of some database query. But a more sophisticated dialogue policy can help a system decide when to answer the user's questions, when to instead ask the user a clarification question, when to make a suggestion, and so on. Finally, dialogue state systems have a **natural language generation** component. In GUS, the sentences that the generator produced were all from pre-written templates. But a more sophisticated generation component can condition on the exact context to produce turns that seem much more natural.

As of the time of this writing, most commercial system are architectural hybrids, based on GUS architecture augmented with some dialogue-state components, but there are a wide variety of dialogue-state systems being developed in research labs.

26.4.1 Dialogue Acts

dialogue acts

Dialogue-state systems make use of **dialogue acts**. Dialogue acts represent the interactive function of the turn or sentence, combining the idea of speech acts and grounding into a single representation. Different types of dialogue systems require labeling different kinds of acts, and so the tagset—defining what a dialogue act is exactly— tends to be designed for particular tasks.

Figure 26.12 shows a tagset for a restaurant recommendation system, and Fig. 26.13 shows these tags labeling a sample dialogue from the HIS system (Young et al., 2010). This example also shows the content of each dialogue acts, which are the slot fillers being communicated. So the user might INFORM the system that they want Italian food near a museum, or CONFIRM with the system that the price is reasonable.

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

Figure 26.12 Dialogue acts used by the HIS restaurant recommendation system of Young et al. (2010). The Sys and User columns indicate which acts are valid as system outputs and user inputs, respectively.

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

Figure 26.13 A sample dialogue from the HIS System of Young et al. (2010) using the dialogue acts in Fig. 26.12.

26.4.2 Slot Filling

The task of slot-filling, and the simpler tasks of domain and intent classification, are special cases of the task of supervised semantic parsing discussed in Chapter 17, in which we have a training set that associates each sentence with the correct set of slots, domain, and intent.

A simple method is to train a a sequence model to map from input words representation to slot fillers, domain and intent. For example given the sentence:

```
I want to fly to San Francisco on Monday afternoon please
```

we compute a sentence representation, for example by passing the sentence through a contextual embedding network like BERT. The resulting sentence representation can be passed through a feedforward layer and then a simple 1-of-N classifier to determine that the domain is AIRLINE and and the intent is SHOWFLIGHT.

IOB Our training data is sentences paired with sequences of **IOB** labels:

```
O O    O  O   O  B-DES I-DES     O  B-DEPTIME I-DEPTIME  O
I want to fly to San   Francisco on Monday    afternoon  please
```

Recall from Chapter 18 that in IOB tagging we introduce a tag for the beginning (B) and inside (I) of each slot label, and one for tokens outside (O) any slot label. The number of tags is thus $2n + 1$ tags, where n is the number of slots.

Fig. 26.14 shows the architecture. The input is a series of words $w_1...w_n$, which is passed through a contextual embedding model to get contextual word representations. This is followed by a feedforward layer and a softmax at each token position over possible IOB tags, with the output is a series of IOB tags $s_1...s_n$. We can also combine the domain-classification and intent-extraction tasks with slot-filling simply by adding a domain concatenated with an intent as the desired output for the final EOS token.

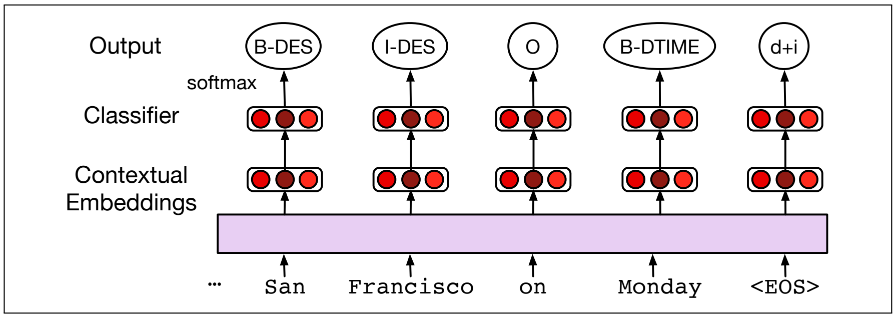

Figure 26.14 A simple architecture for slot filling, mapping the words in the input through contextual embeddings like BERT to an output classifier layer (which can be linear or something more complex), followed by softmax to generate a series of IOB tags (and including a final state consisting of a domain concatenated with an intent).

Once the sequence labeler has tagged the user utterance, a filler string can be extracted for each slot from the tags (e.g., "San Francisco"), and these word strings can then be normalized to the correct form in the ontology (perhaps the airport code 'SFO'). This normalization can take place by using homonym dictionaries (specifying, for example, that SF, SFO, and San Francisco are the same place).

In industrial contexts, machine learning-based systems for slot-filling are often bootstrapped from GUS-style rule-based systems in a semi-supervised learning manner. A rule-based system is first built for the domain, and a test-set is carefully labeled. As new user utterances come in, they are paired with the labeling provided by the rule-based system to create training tuples. A classifier can then be trained on these tuples, using the test-set to test the performance of the classifier against the rule-based system. Some heuristics can be used to eliminate errorful training tuples, with the goal of increasing precision. As sufficient training samples become available the resulting classifier can often outperform the original rule-based system (Suendermann et al., 2009), although rule-based systems may still remain higher-precision for dealing with complex cases like negation.

26.4.3 Dialogue State Tracking

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

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

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

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

has the structure

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

Dialogue act interpretation—in this example choosing `inform` from the set of dialogue acts for this task—is done by supervised classification trained on hand-labeled dialog acts, predicting the dialogue act tag based on embeddings representing the current input sentence and the prior dialogue acts.

The simplest dialogue state tracker might just take the output of a slot-filling sequence-model (Section 26.4.2) after each sentence. Alternatively, a more complex model can make use of the reading-comprehension architectures from Chapter 25. For example the model of Gao et al. (2019) trains a classifier for each slot to decide whether its value is being changed in the current sentence or should be carried over from the previous sentences. If the slot value is being changed, a span-prediction model is used to predict the start and end of the span with the slot filler.

A special case: detecting correction acts

Some dialogue acts are important because of their implications for dialogue control. If a dialogue system misrecognizes or misunderstands an utterance, the user will generally correct the error by repeating or reformulating the utterance. Detecting these **user correction acts** is therefore quite important. Ironically, it turns out that corrections are actually *harder* to recognize than normal sentences! In fact, corrections in one early dialogue system (the TOOT system) had double the ASR word error rate of non-corrections (Swerts et al., 2000)! One reason for this is that speakers sometimes use a specific prosodic style for corrections called **hyperarticulation**, in which the utterance contains exaggerated energy, duration, or F0 contours, such as *I said BAL-TI-MORE, not Boston* (Wade et al. 1992, Levow 1998, Hirschberg et al. 2001). Even when they are not hyperarticulating, users who are frustrated seem to speak in a way that is harder for speech recognizers (Goldberg et al., 2003).

user correction acts

hyperarticulation

What are the characteristics of these corrections? User corrections tend to be either exact repetitions or repetitions with one or more words omitted, although they may also be paraphrases of the original utterance. (Swerts et al., 2000). Detecting these reformulations or correction acts can be part of the general dialogue act detection classifier. Alternatively, because the cues to these acts tend to appear in different ways than for simple acts (like INFORM or *request*, we can make use of features orthogonal to simple contextual embedding features; some typical features are shown below (Levow 1998, Litman et al. 1999, Hirschberg et al. 2001, Bulyko et al. 2005, Awadallah et al. 2015):

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

26.4.4 Dialogue Policy

dialogue policy

The goal of the **dialogue policy** is to decide what action the system should take next, that is, what dialogue act to generate.

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

$$\hat{A}_i = \underset{A_i \in A}{\operatorname{argmax}} P(A_i|(A_1,U_1,...,A_{i-1},U_{i-1}) \tag{26.6}$$

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

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

$$\hat{A}_i = \underset{A_i \in A}{\operatorname{argmax}} P(A_i|\text{Frame}_{i-1},A_{i-1},U_{i-1}) \tag{26.7}$$

These probabilities can be estimated by a neural classifier using neural representations of the slot fillers (for example as spans) and the utterances (for example as sentence embeddings computed over contextual embeddings)

More sophisticated models train the policy via **reinforcement learning**. To decide which action to take, a reinforcement learning system gets a reward at the end of the dialogue, and uses that reward to train a policy to take actions. For example in the movie-recommendation dialogue system of Fazel-Zarandi et al. (2017), the action space has only three actions: EXECUTE, CONFIRM, and ELICIT. The EXECUTE sends a query to the database and answers the user's question, CONFIRM clarifies the intent or slot with the users (e.g., "Do you want movies directed by Christopher Nolan?") while ELICIT asks the user for missing information (e.g., "Which movie are you talking about?"). The system gets a large positive reward if the dialogue system terminates with the correct slot representation at the end, a large negative reward if the slots are wrong, and a small negative reward for confirmation and elicitation questions to keep the system from re-confirming everything.

Policy Example: Confirmation and Rejection

Modern dialogue systems often make mistakes. 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 two methods: **confirming** understandings with the user and **rejecting** utterances that the system is likely to have misunderstood.

explicit confirmation

As we saw in the prior section, most systems introduce particular strategies and actions related to confirmation and rejection. When using the **explicit confirmation** strategy, a system asks the user a direct question to confirm the system's understanding, like the two examples below in which the system asks a (boldface) yes-no confirmation questions:

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

implicit confirmation

When using the **implicit confirmation** strategy, a system instead can demonstrate its understanding as a **grounding** strategy, for example repeating back the system's understanding as part of asking the next question, as in the two examples below:

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

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

rejection

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

progressive prompting

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

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

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

rapid reprompting

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

It is common to use rich features other than just the dialogue state representation to make policy decisions. For example, the **confidence** that the ASR system assigns to an utterance can be used by explicitly confirming low-confidence sentences. As we'll see in page **??**, confidence is a metric that the speech recognizer can assign to its transcription of a sentence to indicate how confident it is in that transcription. Confidence is often computed from the acoustic log-likelihood of the utterance (greater probability means higher confidence), but prosodic features can also be used in confidence prediction. For example, utterances with large F0 excursions or longer durations, or those preceded by longer pauses, are likely to be misrecognized (Litman et al., 2000).

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

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

26.4.5 Natural language generation in the dialogue-state model

Finally, once the policy has decided what speech act to generate, the natural language generation component needs to generate the text of a response to the user.

content planning
sentence realization

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

Here we'll assume content planning has been done by the dialogue policy, which has chosen the dialogue act to generate, and chosen some attributes (slots and values) that the planner wants to say to the user (either to give the user the answer, or as part of a confirmation strategy).

Fig. 26.15 shows some sample input/outputs for the sentence realization phase. In the first example, the content planner has chosen the dialogue act RECOMMEND and some particular slots (name, neighborhood, cuisine) and their fillers. The goal of the sentence realizer is to generate a sentence like lines 1 or 2 shown in the figure, by training on many such examples of representation/sentence pairs from a large corpus of labeled dialogues.

Training data is hard to come by; we are unlikely to see every possible restaurants with every possible attribute in many possible differently worded sentences. There-

	`recommend(restaurant name= Au Midi, neighborhood = midtown, cuisine = french`
1	Au Midi is in Midtown and serves French food.
2	There is a French restaurant in Midtown called Au Midi.
	`recommend(restaurant name= Loch Fyne, neighborhood = city centre, cuisine = seafood)`
3	Loch Fyne is in the City Center and serves seafood food.
4	There is a seafood restaurant in the City Centre called Loch Fyne.

Figure 26.15 Two examples of inputs to the sentence realization phase of NLG, showing the dialogue act and attributes prespecified by the content planner. Line 1-2 and 3-4 show different possible output sentences to be generated by the sentence realizer. From the restaurant recommendation system of Nayak et al. (2017).

delexicalization

fore it is common in sentence realization to increase the generality of the training examples by **delexicalization**. Delexicalization is the process of replacing specific words in the training set that represent slot values with a generic placeholder token representing the slot. Fig. 26.16 shows the result of delexicalizing the training sentences in Fig. 26.15.

	`recommend(restaurant name= Au Midi, neighborhood = midtown, cuisine = french`
1	`restaurant_name` is in `neighborhood` and serves `cuisine` food.
2	There is a `cuisine` restaurant in `neighborhood` called `restaurant_name`.

Figure 26.16 Delexicalized sentences that can be used for generating many different relexicalized sentences. From the restaurant recommendation system of Nayak et al. (2017).

Mapping from frames to delexicalized sentences is generally done by encoder decoder models (Wen et al. 2015a, Wen et al. 2015b, Mrkšić et al. 2017, inter alia), trained on large hand-labeled corpora of task-oriented dialogue (Budzianowski et al., 2018). The input to the encoder is a sequence of tokens x_t that represent the dialogue act and its arguments. Thus the attribute/value pairs decor:decent, service:good, cuisine: null might be represented as a flat sequence of tokens, each mapped to a learned embedding w_t, as shown in Fig. 26.17.

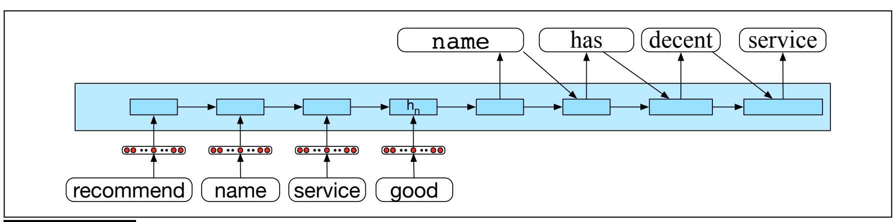

Figure 26.17 An encoder decoder sentence realizer mapping slots/fillers to English.

The encoder reads all the input slot/value representations, produces a context vector that is used as input to the lexical decoder, which generates an English sentence Let's suppose in this case we produce the following (delexicalized) sentence:

restaurant_name has decent service

relexicalize

Then once we've generated the delexicalized string, we can use the input frame from the content planner to **relexicalize** (fill in the exact restaurant or neighborhood or cuisine). This sentence is then relexicalized from the true values in the input frame, resulting in the final sentence:

Au Midi has decent service

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

Generating Clarification Questions

clarification questions

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

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

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

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

26.5 Evaluating Dialogue Systems

Evaluation is crucial in dialogue system design. If the task is unambiguous, we can simply measure absolute task success (did the system book the right plane flight, or put the right event on the calendar).

To get a more fine-grained idea of user happiness, we can compute a *user satisfaction rating*, having users interact with a dialogue system to perform a task and then having them complete a questionnaire. For example, Fig. 26.18 shows sample multiple-choice questions (Walker et al., 2001); responses are mapped into the range of 1 to 5, and then averaged over all questions to get a total user satisfaction rating.

It is often economically infeasible to run complete user satisfaction studies after every change in a system. For this reason, it is useful to have performance evaluation heuristics that correlate well with human satisfaction. A number of such factors and heuristics have been studied, often grouped into two kinds of criteria: how well the system allows users to accomplish their goals (maximizing task success) with the fewest problems (minimizing costs):

Task completion success:

Task success can be measured by evaluating the correctness of the total solution. For a frame-based architecture, this might be **slot error rate** the percentage of slots that were filled with the correct values:

$$\text{Slot Error Rate for a Sentence} = \frac{\text{\# of inserted/deleted/subsituted slots}}{\text{\# of total reference slots for sentence}} \quad (26.8)$$

For example consider a system given this sentence:

(26.9) Make an appointment with Chris at 10:30 in Gates 104

which extracted the following candidate slot structure:

Slot	Filler
PERSON	Chris
TIME	11:30 a.m.
ROOM	Gates 104

Here the slot error rate is 1/3, since the TIME is wrong. Instead of error rate, slot precision, recall, and F-score can also be used.

Interestingly, sometimes the user's *perception* of whether they completed the task is a better predictor of user satisfaction than the actual task completion success. (Walker et al., 2001).

A perhaps more important, although less fine-grained, measure of success is an extrinsic metric like **task error rate**. In this case, the task error rate would quantify how often the correct meeting was added to the calendar at the end of the interaction.

Efficiency cost:

Efficiency costs are measures of the system's efficiency at helping users. This can be measured by 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 interactions 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. 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. The most important is the **slot error rate** described above, but other components include the inappropriateness (verbose or ambiguous) of the system's questions, answers, and error messages or the correctness of each question, answer, or error message (Zue et al. 1989, Polifroni et al. 1992).

26.6 Dialogue System Design

The user plays a more important role in dialogue systems than in most other areas of speech and language processing, and thus the study of dialogue systems is closely linked with the field of Human-Computer Interaction (HCI). The design of dialogue strategies, prompts, and error messages, is often called **voice user interface** design, and generally follows **user-centered design** principles (Gould and Lewis, 1985):

voice user interface

1. Study the user and task: Understand the potential users and the nature of the task by interviews with users, investigation of similar systems, and study of related human-human dialogues.

Wizard-of-Oz system

2. Build simulations and prototypes: A crucial tool in building dialogue systems is the **Wizard-of-Oz system**. In wizard systems, the users interact with what they think is a software agent but is in fact a human "wizard" disguised by a software interface (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 or screen.

A Wizard-of-Oz system can be used to test out an architecture before implementation; only the interface software and databases need to be in place. The wizard gets input from the user, has a graphical interface to a database to run sample queries based on the user utterance, and then has a way to output sentences, either by typing them or by some combination of selecting from a menu and typing. The wizard's linguistic output can be disguised by a text-to-speech system or, more frequently, by using text-only interactions.

The results of a Wizard-of-Oz system can also be used as training data to train a pilot dialogue system. While Wizard-of-Oz systems are very commonly used, they are not a perfect simulation; it is difficult for the wizard to exactly simulate the errors, limitations, or time constraints of a real system; results of wizard studies are thus somewhat idealized, but still can provide a useful first idea of the domain issues.

3. Iteratively test the design on users: An iterative design cycle with embedded user testing is essential in system design (Nielsen 1992, Cole et al. 1997, Yankelovich et al. 1995, Landauer 1995). For example in a famous anecdote in dialogue design history, an early dialogue system required the user to press a key to interrupt the system Stifelman et al. (1993). But user testing showed users barged in, which led to a redesign of the system to recognize overlapped speech. The iterative method is also important for designing prompts that cause the user to respond in normative ways.

There are a number of good books on conversational interface design (Cohen et al. 2004, Harris 2005, Pearl 2017).

26.6.1 Ethical Issues in Dialogue System Design

Ethical issues have long been understood to be crucial in the design of artificial agents, predating the conversational agent itself. Mary Shelley's classic discussion of the problems of creating agents without a consideration of ethical and humanistic concerns lies at the heart of her novel *Frankenstein*. One important ethical issue has to do with bias. As we discussed in Section **??**, machine learning systems of any kind tend to replicate biases that occurred in the training data. This is especially relevant for chatbots, since

both IR-based and neural transduction architectures are designed to respond by approximating the responses in the training data.

Tay

A well-publicized instance of this occurred with Microsoft's 2016 **Tay** chatbot, which was taken offline 16 hours after it went live, when it began posting messages with racial slurs, conspiracy theories, and personal attacks. Tay had learned these biases and actions from its training data, including from users who seemed to be purposely teaching it to repeat this kind of language (Neff and Nagy, 2016).

Henderson et al. (2017) examined some standard dialogue datasets (drawn from Twitter, Reddit, or movie dialogues) used to train corpus-based chatbots, measuring bias (Hutto et al., 2015) and offensive and hate speech (Davidson et al., 2017). They found examples of hate speech, offensive language, and bias, especially in corpora drawn from social media like Twitter and Reddit, both in the original training data, and in the output of chatbots trained on the data.

Another important ethical issue is privacy. Already in the first days of ELIZA, Weizenbaum pointed out the privacy implications of people's revelations to the chatbot. Henderson et al. (2017) point out that home dialogue agents may accidentally record a user revealing private information (e.g. "Computer, turn on the lights –answers the phone –Hi, yes, my password is..."), which may then be used to train a conversational model. They showed that when an encoder-decoder dialogue model is trained on a standard corpus augmented with training keypairs representing private data (e.g. the keyphrase "social security number" followed by a number), an adversary who gave the keyphrase was able to recover the secret information with nearly 100% accuracy. Chatbots that are trained on transcripts of human-human or human-machine conversation must therefore anonymize personally identifiable information. It is the role of the Institutional Review Board (**IRB**) at a researcher's institution to review research proposals for such ethical issues.

IRB

Finally, chatbots raise important issues of gender equality. Current chatbots are overwhelmingly given female names, likely perpetuating the stereotype of a subservient female servant (Paolino, 2017). And when users use sexually harassing language, most commercial chatbots evade or give positive responses rather than responding in clear negative ways (Fessler, 2017).

26.7 Summary

Conversational agents are crucial speech and language processing applications that are already widely used commercially.

- In human dialogue, speaking is a kind of action; these acts are referred to as speech acts or dialogue acts. Speakers also attempt to achieve **common ground** by acknowledging that they have understand each other. Conversation also is characterized by turn structure and dialogue structure.
- Chatbots are conversational agents designed to mimic the appearance of informal human conversation. Rule-based chatbots like ELIZA and its modern descendants use rules to map user sentences into system responses. Corpus-based chatbots mine logs of human conversation to learn to automatically map user sentences into system responses.
- For task-based dialogue, most commercial dialogue systems use the GUS or

frame-based architecture, in which the designer specifies frames consisting of slots that the system must fill by asking the user.

- The **dialogue-state** architecture augments the GUS frame-and-slot architecture with richer representations and more sophisticated algorithms for keeping track of user's dialogue acts, **policies** for generating its own dialogue acts, and a natural language component.
- Dialogue systems are a kind of human-computer interaction, and general HCI principles apply in their design, including the role of the user, simulations such as Wizard-of-Oz systems, and the importance of iterative design and testing on real users.

Bibliographical and Historical Notes

The earliest conversational systems were chatbots like ELIZA (Weizenbaum, 1966) and PARRY (Colby et al., 1971). ELIZA had a widespread influence on popular perceptions of artificial intelligence, and brought up some of the first ethical questions in natural language processing —such as the issues of privacy we discussed above as well the role of algorithms in decision-making— leading its creator Joseph Weizenbaum to fight for social responsibility in AI and computer science in general.

Another early system, the GUS system (Bobrow et al., 1977) had by the late 1970s established the main frame-based paradigm that became the dominant industrial paradigm for dialogue systems for over 30 years.

In the 1990s, stochastic models that had first been applied to natural language understanding began to be applied to dialogue slot filling (Miller et al. 1994, Pieraccini et al. 1991).

By around 2010 the GUS architecture finally began to be widely used commercially in phone-based dialogue systems like Apple's SIRI (Bellegarda, 2013) and other digital assistants.

The rise of the web and online chatbots brought new interest in chatbots and gave rise to corpus-based chatbot architectures around the turn of the century, first using information retrieval models and then in the 2010s, after the rise of deep learning, with sequence-to-sequence models.

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

conversation analysis

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

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

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

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

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

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

BDI

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

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

Affect has played an important role in dialogue systems since its earliest days. In more recent work Mairesse and Walker (2008) showed that conversational agents are received better by users if they match users' personality expectations. Rashkin et al. (2019) introduced the EMPATHETICDIALOGUES dataset of 25k conversations grounded in emotional situations, and (Lin et al., 2019) used mixtures of empathetic listeners (MoEL), each optimized to react to particular emotions, to generate empathetic responses.

[TBD: History of deep reinforcement learning here.] [TBD: surveys: Tur and De Mori (2011), Gao et al. (2019)]

[TBD: more history here on dialogue state tracking, NLG, end-to-end neural systems, etc]

Exercises

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

dispreferred response

26.2 A **dispreferred response** is a response that has the potential to make a person uncomfortable or embarrassed in the conversational context; the most common example dispreferred responses is turning down a request. People signal their discomfort with having to say no with surface cues (like the word *well*), or via 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.

26.3 When asked a question to which they aren't sure they know the answer, people display their lack of confidence by 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.

26.4 Implement a small air-travel help system based on text input. Your system should get constraints from users 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.

26.5 Test your email-reading system on some potential users. Choose some of the metrics described in Section 26.5 and evaluate your system.

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

Artstein, R., Gandhe, S., Gerten, J., Leuski, A., and Traum, D. (2009). Semi-formal evaluation of conversational characters. In *Languages: From Formal to Natural*, 22–35. Springer.

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

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

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

Baum, L. F. (1900). *The Wizard of Oz*. Available at Project Gutenberg.

Bellegarda, J. R. (2013). Natural language technology in mobile devices: Two grounding frameworks. In *Mobile Speech and Advanced Natural Language Solutions*, 185–196. Springer.

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.

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

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *CoNLL-16*, 10–21.

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

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018). MultiWOZ - a large-scale multi-domain wizard-of-Oz dataset for task-oriented dialogue modelling. In *EMNLP 2018*, 5016–5026.

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

Carpenter, R. (2017). Cleverbot. http://www.cleverbot.com, accessed 2017.

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, 12–17. AAAI Press.

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.

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

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

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

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

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

Colby, K. M., Hilf, F. D., Weber, S., and Kraemer, H. C. (1972). Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, *3*, 199–221.

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.

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*, 34–39. AAAI Press.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *ICWSM 2017*.

Fazel-Zarandi, M., Li, S.-W., Cao, J., Casale, J., Henderson, P., Whitney, D., and Geramifard, A. (2017). Learning robust dialog policies in noisy environments. In *Conversational AI Workshop (NIPS)*.

Fessler, L. (2017). We tested bots like Siri and Alexa to see who would stand up to sexual harassment. In *Quartz*. Feb 22, 2017. https://qz.com/911681/.

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

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

Forchini, P. (2013). Using movie corpora to explore spoken American English: Evidence from multi-dimensional analysis. In Bamford, J., Cavalieri, S., and Diani, G. (Eds.), *Variation and Change in Spoken and Written Discourse: Perspectives from corpus linguistics*, 123–136. Benjamins.

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

Gao, S., Sethi, A., Aggarwal, S., Chung, T., and Hakkani-Tür, D. (2019). Dialog state tracking: A neural reading comprehension approach. arXiv preprint arXiv:1908.01946.

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

Godfrey, J., Holliman, E., and McDaniel, J. (1992). SWITCHBOARD: Telephone speech corpus for research and development. In *ICASSP-92*, 517–520.

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

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

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

Gopalakrishnan, K., Hedayatnia, B., Chen, Q., Gottardi, A., Kwatra, S., Venkatesh, A., Gabriel, R., and Hakkani-Tür, D. (2019). Topical-chat: Towards knowledge-grounded open-domain conversations..

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

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

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

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

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

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*, 417–444. MIT Press.

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*, 163–200. Lawrence Erlbaum.

Harris, R. A. (2005). *Voice Interaction Design: Crafting the New Conversational Speech Systems*. Morgan Kaufmann.

Henderson, P., Sinha, K., Angelard-Gontier, N., Ke, N. R., Fried, G., Lowe, R., and Pineau, J. (2017). Ethical challenges in data-driven dialogue systems. In *AAAI/ACM AI Ethics and Society Conference*.

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

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

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

Hutto, C. J., Folds, D., and Appling, S. (2015). Computationally detecting and quantifying the degree of bias in sentence-level text of news stories. In *HUSO 2015: The First International Conference on Human and Social Analytics*.

Isbell, C. L., Kearns, M., Kormann, D., Singh, S., and Stone, P. (2000). Cobot in LambdaMOO: A social statistics agent. In *AAAI/IAAI*, 36–41.

Jafarpour, S., Burges, C. J. C., and Ritter, A. (2009). Filter, rank, and transfer the knowledge: Learning to chat. In *NIPS Workshop on Advances in Ranking*.

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

Kannan, A. and Vinyals, O. (2016). Adversarial evaluation of dialogue models. In *NIPS 2016 Workshop on Adversarial Training*.

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

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

Leuski, A. and Traum, D. (2011). NPCEditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine*, *32*(2), 42–56.

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

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

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

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

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016a). A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016*.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016b). Deep reinforcement learning for dialogue generation. In *EMNLP 2016*.

Li, J., Monroe, W., Shi, T., Ritter, A., and Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. In *EMNLP 2017*.

Lin, Z., Madotto, A., Shin, J., Xu, P., and Fung, P. (2019). MoEL: Mixture of empathetic listeners. http://arxiv.org/abs/1908.07687.

Lison, P. and Tiedemann, J. (2016). Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC-16*.

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. (1987). A plan recognition model for subdialogues in conversation. *Cognitive Science*, *11*, 163–200.

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

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

Liu, C.-W., Lowe, R. T., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP 2016*.

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

Lowe, R. T., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y., and Pineau, J. (2017a). Towards an automatic Turing test: Learning to evaluate dialogue responses. In *ACL 2017*.

Lowe, R. T., Pow, N., Serban, I. V., Charlin, L., Liu, C.-W., and Pineau, J. (2017b). Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, *8*(1), 31–65.

Mairesse, F. and Walker, M. A. (2008). Trainable generation of big-five personality styles through data-driven parameter estimation. In *ACL-08*.

Miller, S., Bobrow, R. J., Ingria, R., and Schwartz, R. (1994). Hidden understanding models of natural language. In *ACL-94*, 25–32.

Mrkšić, N., Ó Séaghdha, D., Wen, T.-H., Thomson, B., and Young, S. (2017). Neural belief tracker: Data-driven dialogue state tracking. In *ACL 2017*, 1777–1788.

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.

Nayak, N., Hakkani-Tür, D., Walker, M. A., and Heck, L. P. (2017). To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation.. In *INTERSPEECH-17*, 3339–3343.

Neff, G. and Nagy, P. (2016). Talking to bots: Symbiotic agency and the case of Tay. *International Journal of Communication*, *10*, 4915–4931.

Nielsen, J. (1992). The usability engineering life cycle. *IEEE Computer*, *25*(3), 12–22.

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

Paolino, J. (2017). Google Home vs Alexa: Two simple user experience design gestures that delighted a female user. In *Medium*. Jan 4, 2017. https://medium.com/startup-grind/google-home-vs-alexa-56e26f69ac77.

Pearl, C. (2017). *Designing Voice User Interfaces: Principles of Conversational Experiences*. O'Reilly.

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*, 121–124.

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

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

Rashkin, H., Smith, E. M., Li, M., and Boureau, Y.-L. (2019). Towards empathetic open-domain conversation models: A new benchmark and dataset. In *ACL 2019*, 5370–5381.

Ritter, A., Cherry, C., and Dolan, B. (2010). Unsupervised modeling of twitter conversations. In *HLT-NAACL*.

Ritter, A., Cherry, C., and Dolan, B. (2011). Data-driven response generation in social media. In *EMNLP-11*, 583–593.

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

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

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

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

Serban, I. V., Lowe, R., Henderson, P., Charlin, L., and Pineau, J. (2018). A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, *9*(1), 1–49.

Shang, L., Lu, Z., and Li, H. (2015). Neural responding machine for short-text conversation. In *ACL 2015*, 1577–1586.

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

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

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

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. In *NAACL HLT 2015*, 196–205.

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

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*, 179–186.

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

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

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

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

Suendermann, D., Evanini, K., Liscombe, J., Hunter, P., Dayanidhi, K., and Pieraccini, R. (2009). From rule-based to statistical grammars: Continuous improvement of large-scale spoken dialog systems. In *ICASSP-09*, 4713–4716.

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

Tur, G. and De Mori, R. (2011). *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

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

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

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

Walker, M. A., Kamm, C. A., and Litman, D. J. (2001). Towards developing general models of usability with PARADISE. *Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems*, *6*(3), 363–377.

Walker, M. A. and Whittaker, S. (1990). Mixed initiative in dialogue: An investigation into discourse segmentation. In *ACL-90*, 70–78.

Wang, H., Lu, Z., Li, H., and Chen, E. (2013). A dataset for research on short-text conversations.. In *EMNLP 2013*, 935–945.

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

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

Weizenbaum, J. (1966). ELIZA – A computer program for the study of natural language communication between man and machine. *CACM*, *9*(1), 36–45.

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

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

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

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

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

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

Yan, Z., Duan, N., Bao, J.-W., Chen, P., Zhou, M., Li, Z., and Zhou, J. (2016). DocChat: An information retrieval approach for chatbot engines using unstructured documents. In *ACL 2016*.

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*, 369–376.

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

Zhou, L., Gao, J., Li, D., and Shum, H.-Y. (2018). The design and implementation of XiaoIce, an empathetic social chatbot..

Zue, V. W., 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*, 160–167.