Introduction to Chatbots and Dialogue Systems
Conversational Agents
(aka Dialogue Systems aka Dialogue Agents aka Chatbots)

Personal Assistants on phones or other devices
SIRI, Alexa, Cortana, Google Assistant
Playing music, setting timers and clocks
Chatting for fun
Booking travel reservations
Clinical uses for mental health
Two kind of conversational agents

1. Chatbots
   - mimic informal human chatting
   - for fun, or even for therapy

2. (Task-based) Dialogue Agents
   - interfaces to personal assistants
   - cars, robots, appliances
   - booking flights or restaurants
Chatbot Architectures

Rule-based
1. Pattern-action rules (ELIZA)
2. + A mental model (PARRY):
   The first system to pass the Turing Test!

Corpus-based
3. Information Retrieval (Xiaolce)
4. Neural encoder-decoder (BlenderBot)
Will you sing me a song?

Sure, what do you want to be sung to? I can sing you a song about baking.

Yes, sing me a song about baking!

Do you want me to sing it to the tune of “Sing Along with Mitch”?

Yes, I would love that.

OK, I’ll try to get it to you by the end of the day. What else do you like to do?

No, sing it to me now!

OK, give me a few days. I’m a teacher, so I don’t have much free time.

Can you at least tell me what the lyrics are?

Sure, it goes like this: “Baking is my favorite thing to do in the world.”

How does the chorus go?

It goes: BAKING IS MY FAVORITE THING TO DO IN THE WORLD!

Wow, sounds like a bop.

I know, right? It’s so catchy. I can’t get it out of my head.
These systems often have an entertainment value, such as Facebook's Blender-Bot, a neural chatbot capable of carrying on the kinds of conversations shown in Figure 24.2, or Microsoft's XiaoIce system (Zhou et al., 2020), a system that chats with people on text messaging platforms (Fig. 24.3).

Sure, what do you want to be sung to? I can sing you a song about baking.

Will you sing me a song?

Yes, sing me a song about baking!

Do you want me to sign it to the tune of "Sing Along with Mitch"?

Yes, I would love that.

OK, I'll try to get it to you by the end of the day. What else do you like to do?

No, sing it to me now!

OK, give me a few days. I'm a teacher, so I don't have much free time.

Can you at least tell me what the lyrics are?

Sure, it goes like this: "Baking is my favorite thing to do in the world."

How does the chorus go?

It goes: BAKING IS MY FAVORITE THING TO DO IN THE WORLD!

Wow, sounds like a bop.

I know, right? It's so catchy. I can't get it out of my head.

Figure 24.2 A conversation between a user (left) and BlenderBot (Roller et al., 2020).
Task-based dialogue agents

"Task-based" or "goal-based" dialogue agents

- Systems that have the goal of helping a user solve a task
  - Setting a timer
  - Making a travel reservation
  - Playing a song
  - Buying a product

Architecture:

- Frames with slots and values
- A knowledge structure representing user intentions
The Frame

A set of **slots**, to be filled with information of a given **type**
Each associated with a **question** to the user

<table>
<thead>
<tr>
<th>Slot</th>
<th>Type</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIGIN</td>
<td>city</td>
<td>&quot;What city are you leaving from?&quot;</td>
</tr>
<tr>
<td>DEST</td>
<td>city</td>
<td>&quot;Where are you going?&quot;</td>
</tr>
<tr>
<td>DEP DATE</td>
<td>date</td>
<td>&quot;What day would you like to leave?&quot;</td>
</tr>
<tr>
<td>DEP TIME</td>
<td>time</td>
<td>&quot;What time would you like to leave?&quot;</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>line</td>
<td>&quot;What is your preferred airline?&quot;</td>
</tr>
</tbody>
</table>
Chatbots and Dialogue Systems

Introduction to Chatbots and Dialogue Systems
Properties of Human Conversation
A telephone conversation between a human travel agent (A) and a human client (C)
Properties of Human Conversation

Turns

• We call each contribution a "turn"
• As if conversation was the kind of game where everyone takes turns.
C1: ...I need to travel in May.
A2: And, what day in May did you want to travel?
C3: OK uh I need to be there for a meeting that’s from the 12th to the 15th.
A4: And you’re flying into what city?
C5: Seattle.
A6: And what time would you like to leave Pittsburgh?
C7: Uh hmm I don’t think there’s many options for non-stop.
A8: Right. There’s three non-stops today.
C9: What are they?
A10: 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.
C11: OK I’ll take the 5ish flight on the night before on the 11th.
A12: On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air
flight 115.
C13: OK.
A14: And you said returning on May 15th?
C15: Uh, yeah, at the end of the day.
A16: OK. There’s #two non-stops . . .#
C17: #Act...actually #, what day of the week is the 15th?
A18: It’s a Friday.
C19: Uh hmm. I would consider staying there an extra day til Sunday.
A20: OK...OK. On Sunday I have . . .
Properties of Human Conversation

**Turn-taking issues**
- When to take the floor?
- When to yield the floor?

**Interruptions**
Chapter 24: Chatbots & Dialogue Systems

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 24.2 we'll discuss the three major chatbot architectures: rule-based systems, information retrieval systems, and encoder-decoder generators. In Section 24.3 we turn to task-oriented agents, introducing the frame-based architecture (the GUS architecture) that underlies most task-based systems.

24.1 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. 24.1.

C1: ...I need to travel in May.
A2: And, what day in May did you want to travel?
C3: OK uh I need to be there for a meeting that’s from the 12th to the 15th.
A4: And you’re flying into what city?
C5: Seattle.
A6: And what time would you like to leave Pittsburgh?
C7: Uh hmm I don’t think there’s many options for non-stop.
A8: Right. There’s three non-stops today.
C9: What are they?
A10: 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.
C11: OK I’ll take the 5ish flight on the night before on the 11th.
C13: OK.
A14: And you said returning on May 15th?
C15: Uh, yeah, at the end of the day.
A16: OK. There’s #two non-stops . . .#
C17: #Act...actually  #, what day of the week is the 15th?
A18: It’s a Friday.
C19: Uh hmm. I would consider staying there an extra day til Sunday.
A20: OK...OK. On Sunday I have . . .
Implications for Conversational Agents

Barge-in

- Allowing the user to interrupt

End-pointing

- The task for a speech system of deciding whether the user has stopped talking.
- Very hard, since people often pause in the middle of turns
Language as Action

Each turn in a dialogue is a kind of action
Wittgenstein (1953) and Austin (1962)
### Speech Acts (aka Dialogue Acts)

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

Bach and Harnish (1979)
Speech acts

"Turn up the music!"
  DIRECTIVE

"What day in May do you want to travel?"
  DIRECTIVE

"I need to travel in May"
  CONSTATIVE

Thanks
  ACKNOWLEDGEMENT
Grounding

• Participants in conversation or any joint activity need to establish **common ground**.

• **Principle of closure.** Agents performing an action require evidence, sufficient for current purposes, that they have succeeded in performing it  (Clark 1996, after Norman 1988)

• Speech is an action too! So speakers need to **ground** each other’s utterances.
  • **Grounding:** acknowledging that the hearer has understood
Grounding

• Grounding is relevant for human-machine interaction
  • Why do elevator buttons light up?
Grounding: Establishing Common Ground

A: And you said returning on May 15th?
C: Uh, yeah, at the end of the day.
A: **OK**

C: OK I’ll take the 5ish flight on the night before on the 11th.
A: **On the 11th? OK.**

C: ...I need to travel in May.
A: **And**, what day **in May** did you want to travel?
Grounding is important for computers too!

System: Did you want to review some more of your profile?
User: No.  
System: What’s next?

System: Did you want to review some more of your profile?
User: No.
System: Okay, what’s next?
Conversations have structure

Local structure between adjacent speech acts, from the field of **conversational analysis** (Sacks et al. 1974)

Called **adjacency pairs**:

**QUESTION... ANSWER**

**PROPOSAL... ACCEPTANCE/REJECTION**

**COMPLIMENTS** ("Nice jacket!")... **DOWNPLAYER** ("Oh, this old thing?")
Another kind of structure: Subdialogues

**Correction** subdialogue

**Agent:** OK. There's two non-stops#

**Client:** #Act- actually#, what day of the week is the 15th?

**Agent:** It’s a Friday.

**Client:** Uh hmm. I would consider staying there an extra day til Sunday.

**Agent:** OK...OK. On Sunday I have ...

"Quote" subdialogue
Clarification Subdialogues

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...
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.
Conversational Initiative

Some conversations are controlled by one person

• A reporter interviewing a chef asks questions, and the chef responds.
• This reporter has the **conversational initiative** (Walker and Whittaker 1990)

Most human conversations have **mixed initiative**:

• I lead, then you lead, then I lead.
• Mixed initiative is very hard for NLP systems, which often default to simpler styles that can be frustrating for humans:
  • **User initiative** (user asks or commands, system responds)
  • **System initiative** (system asks user questions to fill out a form, user can't change the direction)
Even harder problems: Inference

Agent: And, what day in May did you want to travel?
Client: OK, uh, I need to be there for a meeting that’s from the 12th to the 15th.
Properties of Human Conversation
Rule-based Chatbots: ELIZA and PARRY
ELIZA: Weizenbaum (1966)

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
...
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

ELIZA's trick: be a Rogerian psychologist

Draw the patient out by reflecting patient’s statements back at them

Rare type of conversation in which one can “assume the pose of knowing almost nothing of the real world”
Rogerian psychologist

Patient: "I went for a long boat ride"
Psychologist: "Tell me about boats"

You don’t assume she didn’t know what a boat is
You assume she had some conversational goal
Chatbots trying to pass the Turing test usually choose some such domain
Eliza pattern/transform rules

\[(0 \ \text{YOU} \ 0 \ \text{ME}) \rightarrow (\text{WHAT MAKES YOU THINK I } 3 \ \text{YOU})]\]

0 means Kleene *
The 3 is the constituent # in pattern

You hate me
WHAT MAKES YOU THINK I HATE YOU
Eliza Rules

Rules are organized by **keywords**
Each keyword has a pattern and a list of possible transform

Keyword: **YOU**
Pattern: (0 YOU 0 ME)
Transforms: (WHAT MAKES YOU THINK I 3 YOU)  
(WHY DO YOU THINK I 3 YOU)

<table>
<thead>
<tr>
<th>keyword</th>
<th>pattern</th>
<th>list of transforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>(K)</td>
<td>((D_1) (R_1, 1) (R_1, 2) \cdots (R_1, m_1))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>((D_2) (R_2, 1) (R_2, 2) \cdots (R_2, m_2))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td></td>
<td>((D_n) (R_n, 1) (R_n, 2) \cdots (R_n, m_n)))</td>
<td></td>
</tr>
</tbody>
</table>


Keywords are ranked from specific to general

**User:** I know everybody laughed at me

“**I**” is very general:

I: (I *) → (YOU SAY YOU 2)

**ELIZA:** YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU

“**Everybody**” is more specific and interesting

Everybody: (Everybody *) → (WHO IN PARTICULAR ARE YOU THINKING OF)

**ELIZA:** WHO IN PARTICULAR ARE YOU THINKING OF?

**Implementation:** keywords stored with their rank

Everybody 5 (list of transformation rules)
I 0 (list of transformation rules)
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)
NONE

PLEASE GO ON
THAT’S VERY INTERESTING
I SEE
Memory

(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))

Whenever “MY” is highest keyword
  ◦ Randomly select a transform on the MEMORY list
  ◦ Apply to sentence
  ◦ Store on a (first-in-first-out) queue

Later, if no keyword matches a sentence
  ◦ Return the top of the MEMORY queue instead
Ethical implications: Anthropomorphism and Privacy

People became deeply emotionally involved with the program. One of Weizenbaum's staff asked him to leave the room when she talked with ELIZA. When he suggested that he might want to store all the ELIZA conversations for later analysis, people immediately pointed out the privacy implications:

- Suggesting that they were having quite private conversations with ELIZA
- Despite knowing that it was just software.
Ethical implications

It worried Weizenbaum that people confided in ELIZA

- Were people misled about how much computers understood?

Turkle studied users of ELIZA and other systems

- Turkle has shown human face-to-face interaction is vital
- But people also develop specific relationships with artifacts
- Some users told her ELIZA was more like a kind of diary, a way to privately explore their thoughts.

Importance of value-sensitive design

PARRY: A computational model of schizophrenia

Another chatbot with a clinical psychology focus


Used to study schizophrenia

Same pattern-response structure as Eliza

But a much richer:

◦ control structure
◦ language understanding capabilities
◦ model of mental state.
◦ variables modeling levels of Anger, Fear, Mistrust
Affect variables

Fear (0-20)  Anger (0-20)  Mistrust (0-15)

Start with all variables low

After each user turn
  ◦ Each user statement can change Fear and Anger
    ◦ E.g., Insults increases Anger, Flattery decreases Anger
    ◦ Mentions of his delusions increase Fear
  ◦ Else if nothing malevolent in input
    ◦ Anger, Fear, Mistrust all drop
Parry's responses depend on mental state

User Input

Modify
Affect variables

condition
excessive fear
excessive anger
mentions delusion
question

response
Escape
Hostility
Fear
answer

...
PARRY passes the Turing test in 1972

The first system to pass a version of the Turing test
Psychiatrists couldn’t distinguish interviews with PARRY from (text transcripts of) interviews with people diagnosed with paranoid schizophrenia

Rule-based Chatbots: ELIZA and PARRY
Corpus-based Chatbots
Two architectures for corpus-based chatbots

Response by retrieval
- Use information retrieval to grab a response (that is appropriate to the context) from some corpus

Response by generation
- Use a language model or encoder-decoder to generate the response given the dialogue context
Corpus-based chatbots require corpora

- Modern corpus-based chatbots are very data-intensive
- They commonly require hundreds of millions or billions of words
What conversations to draw on?

Transcripts of telephone conversations between volunteers
  • Switchboard corpus of American English telephone conversations

Movie dialogue
  • Various corpora of movie subtitles

Hire human crowdworkers to have conversations
  • Topical-Chat 11K crowdsourced conversations on 8 topics
  • EMPATHETICDIALOGUES 25K crowdsourced conversations grounded in a situation where a speaker was feeling a specific emotion

Pseudo-conversations from public posts on social media
  • Drawn from Twitter, Reddit, Weibo (微博), etc.
  • Tend to be noisy; often used just as pre-training.

Crucial to remove personally identifiable information (PII)
Response by retrieval: classic IR method

1. Given a user turn $q$, and a training corpus $C$ of conversation
2. Find in $C$ the turn $r$ that is most similar (tf-idf cosine) to $q$
3. Say $r$

$$\text{response}(q, C) = \underset{r \in C}{\text{argmax}} \frac{q \cdot r}{||q|| \cdot ||r||}$$
Response by retrieval: neural IR method

1. Given a user turn $q$, and a training corpus $C$ of conversation
2. Find in $C$ the turn $r$ that is most similar (BERT dot product) to $q$
3. Say $r$

$$h_q = \text{BERT}_Q(q)[\text{CLS}]$$
$$h_r = \text{BERT}_R(r)[\text{CLS}]$$

$$\text{response}(q,C) = \arg\max_{r \in C} h_q \cdot h_r$$
Response by generation

Think of response production as an encoder-decoder task

Generate each token $r_t$ of the response by conditioning on the encoding of the entire query $q$ and the response so far $r_1...r_{t-1}$

$$\hat{r}_t = \arg\max_{w \in V} P(w|q, r_1...r_{t-1})$$

(a) Response by Retrieval

(b) Response by Generation
The IR-based approach can be extended in various ways, such as by using more sophisticated neural architectures (Humeau et al., 2020), or by using a longer context for the query than just the user's last turn, up to the whole preceding conversation. Information about the user or sentiment or other information can also play a role.

An alternate way to use a corpus to generate dialogue is to think of response production as an encoder-decoder task—transducing from the user's prior turn to the system's turn. We can think of this as a machine learning version of ELIZA; the system learns from a corpus to transduce a question to an answer. Ritter et al. (2011) proposed early on to think of response generation as a kind of translation, and this idea was generalized to the encoder-decoder model roughly contemporaneously by Shang et al. (2015), Vinyals and Le (2015), and Sordoni et al. (2015).

As we saw in Chapter 11, encoder decoder models generate each token $r_t$ of the response by conditioning on the encoding of the entire query $q$ and the response so far $r_1...r_{t-1}$:

$$\hat{r}_t = \arg \max_w P(w|q, r_1...r_{t-1})$$ (24.3)

Fig. 24.6 shows the intuition of the generator and retriever methods for response generation. In the generator architecture, we normally include a longer context, forming the query not just from the user's turn but from the entire conversation-so-far. Fig. 24.7 shows an fleshed-out example.

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
Response by generation

Alternative approach: fine-tune a large language model on conversational data

The Chirpy Cardinal system (Paranjape et al., 2020):
• fine-tunes GPT-2
• on the EMPATHETICDIALOGUES dataset (Rashkin et al., 2019)
“What ho!” I said.
“What ho!” said Motty.
“What ho! What ho!”
After that it seemed rather difficult to go on with the conversation.”

Wodehouse *My Man Jeeves*, 1919
Ongoing research problem: Neural chatbots can get repetitive and boring

A: Where are you going?
B: I’m going to the restroom.
A: See you later.
B: See you later.
A: See you later.
B: See you later.

A: How old are you?
B: I’m 16.
A: 16?
B: I don’t know what you are talking about.
A: You don’t know what you are saying.
B: I don’t know what you are talking about.
A: You don’t know what you are saying.
Response by retrieving and refining knowledge

Can generate responses from informative text rather than dialogue

- To respond to turns like “Tell me something about Beijing”
  - Xiaolce collects sentences from public lectures and news articles.
  - And searches them using IR based on query expansion from user’s turn

- Can augment encoder-decoder model
  - use IR to retrieve passages from Wikipedia
  - concatenate each Wikipedia sentence to the dialogue context with a separator token.
  - Give as encoder context to the encoder-decoder model, which learns to incorporate text into its response
Hybrid Architectures

Chirpy Cardinal (Paranjape et al., 2020) response generation from a series of different generators:

• GPT-2 finetuned on EmpatheticDialogues
• GPT-2 finetuned to paraphrase content from Wikipedia
• Rule-based movie or music generators that produce scripted conversation about a movie or a musician
  • asking the user’s opinion about a movie,
  • giving a fun fact,
  • asking the user their opinion on an actor in the movie.
<table>
<thead>
<tr>
<th>#</th>
<th>Utterance</th>
<th>Entity</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>User</strong>: let’s chat</td>
<td>None</td>
<td><strong>Launch</strong> gives <strong>FORCE_START</strong> response to begin the conversation.</td>
</tr>
<tr>
<td></td>
<td><strong>Bot</strong>: Hi, this is an Alexa Prize Socialbot. I’d love to get to know you a bit better before we chat! Is it all right if I ask for your name? <strong>[Launch]</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>User</strong>: my name is c<strong>r</strong>is</td>
<td>None</td>
<td><strong>Launch</strong> gives a <strong>STRONG_CONTINUE</strong> response that needs a prompt, which is supplied by <strong>Neural Chat</strong> with <strong>FORCE_START</strong> priority (scripted transition).</td>
</tr>
<tr>
<td></td>
<td><strong>Bot</strong>: Well it’s nice to meet you, Chris! I’m excited to chat with you today. <strong>[Launch]</strong> I hope your afternoon is going well. What are your plans for the rest of today? <strong>[Neural Chat]</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>User</strong>: hang out with my friends</td>
<td>None</td>
<td><strong>Neural Chat</strong> generates a GPT-2 response with <strong>STRONG_CONTINUE</strong> priority.</td>
</tr>
<tr>
<td></td>
<td><strong>Bot</strong>: That sounds great, I love spending time with friends. What will you do together? <strong>[Neural Chat]</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>User</strong>: maybe watch a movie</td>
<td>None</td>
<td><strong>Movies</strong> detects the ‘movie’ keyword in the user’s utterance and gives a <strong>FORCE_START</strong> response (overriding <strong>Neural Chat</strong>), setting current entity to <strong>Film</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Bot</strong>: Me too! I love watching movies; I get to learn so much about what the world is like outside of the cloud! Have you seen any movies recently? <strong>[Movies]</strong></td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>User</strong>: i saw the matrix</td>
<td>The Matrix</td>
<td><strong>The Entity Linker</strong> and <strong>Entity Tracker</strong> identify <strong>The Matrix</strong> as the current entity.</td>
</tr>
<tr>
<td></td>
<td><strong>Bot</strong>: Nice! Did you like <strong>The Matrix</strong>? <strong>[Movies]</strong></td>
<td>The Matrix</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>User</strong>: i loved it neo is amazing</td>
<td>The Matrix</td>
<td><strong>Movies</strong> sets current entity to <strong>Keanu Reeves</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Bot</strong>: Oooh, yeah, I agree. Hey, isn’t Keanu Reeves in that movie? What do you think about Keanu Reeves? <strong>[Movies]</strong></td>
<td>Keanu Reeves</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>User</strong>: i want to talk about something else</td>
<td>None</td>
<td><strong>Navigational Intent</strong> detects negative intent, so <strong>Entity Tracker</strong> rejects the current entity.</td>
</tr>
</tbody>
</table>
Chatbots: pro and con

Pro:
- Fun
- Good for narrow, scriptable applications

Cons:
- They don't really understand
- Giving the appearance of understanding may be problematic
- Rule-based chatbots are expensive and brittle
- IR-based chatbots can only mirror training data
- We'll discuss later the case of Microsoft Tay
  - (or, Garbage-in, Garbage-out)

Next steps: integrating chatbot ability into frame-based agents
Corpus-based Chatbots
The Frame-based ("GUS") Dialogue Architecture
Frame-based dialogue agents

Sometimes called "task-based dialogue agents"

- Systems that have the goal of helping a user solve a task like making a travel reservation or buying a product

Architecture:

- First proposed in the GUS system of 1977
- A knowledge structure representing user intentions
- One or more frames (each consisting of slots with values)
The Frame

A set of slots, to be filled with information of a given type
Each associated with a question to the user
Sometimes called a domain ontology

<table>
<thead>
<tr>
<th>Slot</th>
<th>Type</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIGIN</td>
<td>city</td>
<td>&quot;What city are you leaving from?&quot;</td>
</tr>
<tr>
<td>DEST</td>
<td>city</td>
<td>&quot;Where are you going?&quot;</td>
</tr>
<tr>
<td>DEP DATE</td>
<td>date</td>
<td>&quot;What day would you like to leave?&quot;</td>
</tr>
<tr>
<td>DEP TIME</td>
<td>time</td>
<td>&quot;What time would you like to leave?&quot;</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>line</td>
<td>&quot;What is your preferred airline?&quot;</td>
</tr>
</tbody>
</table>
Two basic architectures

The **GUS** architecture

- Sometimes just called "**frame-based**" architecture
- Over 40 years old, but still used in most industrial task-based dialogue agents

The **dialogue-state** architecture

- Extension of GUS
- More common in research systems
- Some aspects making their way into industrial systems
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
Control structure for GUS frame architecture

System asks questions of user, filling any slots that user specifies

User might fill many slots at a time:

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

When frame is filled, do database query
GUS slots have condition-action rules attached

Some rules attached to the DESTINATION slot for the plane booking frame

1. Once the user has specified the destination
   • Enter that city as the default *StayLocation* for the hotel booking frame.

2. Once the user has specified DESTINATION DAY for a short trip
   • Automatically copy as ARRIVAL DAY.
GUS systems have multiple frames

Frames like:

• Car or hotel reservations
• General route information
  • Which airlines fly from Boston to San Francisco?
• Information about airfare practices
  • Do I have to stay a specific number of days to get a decent airfare?

Frame detection:

• System must detect which slot of which frame user is filling
• And switch dialogue control to that frame.
GUS: Natural Language Understanding for filling dialog slots

1. **Domain classification**
   - Asking weather? Booking a flight? Programming alarm clock?

2. **Intent Determination**
   - Find a Movie, Show Flight, Remove Calendar Appt

3. **Slot Filling**
   - Extract the actual slots and fillers
Natural Language Understanding for filling slots

Show me morning flights from Boston to SF on Tuesday.

**DOMAIN:** AIR-TRAVEL  
**INTENT:** SHOW-FLIGHTS  
**ORIGIN-CITY:** Boston  
**ORIGIN-DATE:** Tuesday  
**ORIGIN-TIME:** morning  
**DEST-CITY:** San Francisco
The goal of the natural language understanding component 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 clocks, 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 dialog 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:

Wake me tomorrow at six.

a system might want to build a representation like:

**DOMAIN:** ALARM-CLOCK  
**INTENT:** SET-ALARM  
**ORIGIN-CITY:** Boston  
**ORIGIN-DATE:** Tuesday  
**ORIGIN-TIME:** morning  
**DEST-CITY:** San Francisco

while an utterance like

Wake me tomorrow at six.

should give an intent like this:

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

The task of slot-filling, and the simpler tasks of domain and intent classification, are special cases of the task of semantic parsing discussed in Chapter ??.

Dialogue agents can thus extract slots, domains, and intents from user utterances by applying any of the semantic parsing approaches discussed in that chapter.

The method used in the original GUS system, and still quite common in industrial applications, is to use hand-written rules, often as part of the condition-action rules attached to slots or concepts.

For example we might just define a regular expression consisting of a set strings that map to the SET-ALARM intent:

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

We can build more complex automata that instantiate sets of rules like those discussed in Chapter 20, for example extracting a slot filler by turning a string like

Monday at 2pm

into an object of type `date` with parameters (DAY, MONTH, YEAR, HOURS, MINUTES).

Rule-based systems can be even implemented with full grammars. Research systems like the Phoenix system (Ward and Issar, 1994) consists of large hand-designed semantic grammars with thousands of rules. A semantic grammar is a context-free semantic 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:
How to fill slots?
(1) Rule-based Slot-filling

Write regular expressions or grammar rules

Wake me (up) | set (the|an) alarm | get me up

Do text normalization
Generating responses: template-based generation

A template is a pre-built response string

Templates can be **fixed**:  
"Hello, how can I help you?"

Or have **variables**:  
"What time do you want to leave CITY-ORIG?"  
"Will you return to CITY-ORIG from CITY-DEST?"
Summary: simple frame-based architecture

Like many rule-based approaches

• Positives:
  • High precision
  • Can provide coverage if the domain is narrow

• Negatives:
  • Can be expensive and slow to create rules
  • Can suffer from recall problems
The Frame-based ("GUS") Dialogue Architecture
Chatbots and Dialogue Systems

The Dialogue-State Architecture
Dialogue-State or Belief-State Architecture

A more sophisticated version of the frame-based architecture

• Has dialogue acts, more ML, better generation

The basis for modern research systems

Slowly making its way into industrial systems

• Some aspects (ML for slot-understanding) already widely used industrially
The topic of this paper is the dialog state tracker (DST). The DST takes as input all of the dialog history so far, and outputs its estimate of the current dialog state – for example, in a restaurant information system, the dialog state might indicate the user's preferred price range and cuisine, what information they are seeking such as the phone number of a restaurant, and which concepts have been stated vs. confirmed. Dialog state tracking is difficult because ASR and SLU errors are common, and can cause the system to misunderstand the user. At the same time, state tracking is crucial because the dialog policy relies on the estimated dialog state to choose actions – for example, which restaurants to suggest.

In the literature, numerous methods for dialog state tracking have been proposed. These are covered in detail in Section 3; illustrative examples include hand-crafted rules (Larsson and Traum, 2000; Bohus and Rudnicky, 2003), heuristic scores (Higashinaka et al., 2003), Bayesian networks (Paek and Horvitz, 2000; Williams and Young, 2007), and discriminative models (Bohus and Rudnicky, 2006). Techniques have been fielded which scale to realistically sized dialog problems and operate in real time (Young et al., 2010; Thomson and Young, 2010; Williams, 2010; Mehta et al., 2010). In end-to-end dialog systems, dialog state tracking has been shown to improve overall system performance (Young et al., 2010; Thomson and Young, 2010).

Despite this progress, direct comparisons between methods have not been possible because past studies use different domains and different system components for ASR, SLU, dialog policy, etc. Moreover, there has not been a standard task or methodology for evaluating dialog state tracking. Together these issues have limited progress in this research area.

The Dialog State Tracking Challenge (DSTC) series has provided a first common testbed and evaluation suite for dialog state tracking. Three instances of the DSTC have been run over a three-year period.
Components in a dialogue-state architecture

**NLU:** extracts slot fillers from the user’s utterance using machine learning

**Dialogue state tracker:** maintains the current state of the dialogue (user’s most recent dialogue act, set of slot-filler constraints from user

**Dialogue policy:** decides what the system should do or say next
- GUS policy: ask questions until the frame was full then report back
- More sophisticated: know when to answer questions, when to ask a clarification question, etc.

**NLG:** produce more natural, less templated utterances
## Dialogue Acts

Combine the ideas of **speech acts** and **grounding** into a single representation

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

Young et al., 2010:
## Dialogue Acts

<table>
<thead>
<tr>
<th>Utterance</th>
<th>Dialogue act</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>U:</strong> Hi, I am looking for somewhere to eat.</td>
<td><code>hello(task = find, type=restaurant)</code></td>
</tr>
<tr>
<td><strong>S:</strong> You are looking for a restaurant. What type of food do you like?</td>
<td><code>confreq(type = restaurant, food)</code></td>
</tr>
<tr>
<td><strong>U:</strong> I’d like an Italian somewhere near the museum.</td>
<td><code>inform(food = Italian, near=museum)</code></td>
</tr>
<tr>
<td><strong>S:</strong> Roma is a nice Italian restaurant near the museum.</td>
<td><code>inform(name = &quot;Roma&quot;, type = restaurant, food = Italian, near = museum)</code></td>
</tr>
<tr>
<td><strong>U:</strong> Is it reasonably priced?</td>
<td><code>confirm(pricerange = moderate)</code></td>
</tr>
<tr>
<td><strong>S:</strong> Yes, Roma is in the moderate price range.</td>
<td><code>affirm(name = &quot;Roma&quot;, pricerange = moderate)</code></td>
</tr>
<tr>
<td><strong>U:</strong> What is the phone number?</td>
<td><code>request(phone)</code></td>
</tr>
<tr>
<td><strong>S:</strong> The number of Roma is 385456.</td>
<td><code>inform(name = &quot;Roma&quot;, phone = &quot;385456&quot;)</code></td>
</tr>
<tr>
<td><strong>U:</strong> Ok, thank you goodbye.</td>
<td><code>bye()</code></td>
</tr>
</tbody>
</table>
Slot filling: Machine learning

Machine learning classifiers to map words to semantic frame-fillers

Given a set of labeled sentences

*Input:* "I want to fly to San Francisco on Monday please"

*Output:* 
- **Destination:** SF
- **Depart-time:** Monday

Build a classifier to map from one to the other

Requirements: Lots of labeled data
Slot filling as sequence labeling: BIO tagging

The **BIO tagging** paradigm

Idea: Train a classifier to label each input word with a tag that tells us what slot (if any) it fills

We create a B and I tag for each slot-type
And convert the training data to this format

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

```
0 0 0 0 0 B-DES I-DES 0 B-DEPTIME I-DEPTIME 0
```

```
Recall from Chapter 8 that in BIO 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 \(2^n+1\) tags, where \(n\) is the number of slots.
```
Fig. 24.15 shows the architecture. The input is a series of words $w_1 \ldots w_n$, 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 BIO tags, with the output a series of BIO tags $s_1 \ldots s_n$. 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.

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.

24.4.3 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 Mrković et al. (2017) shows the required output of the dialogue state tracker after each turn:

Can do domain and intent too: e.g., generate the label "AIRLINE_TRAVEL + SEARCH_FLIGHT"
Once we have the BIO tag of the sentence

0 0 0 0 0 B-DES I-DES 0 B-DEPTIME I-DEPTIME 0
I want to fly to San Francisco on Monday afternoon please

• We can extract the filler string for each slot
• And then normalize it to the correct form in the ontology
• Like "SFO" for San Francisco
• Using homonym dictionaries (SF=SFO=San Francisco)
The task of dialogue state tracking

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

Dialogue state tracking

I'd like Cantonese food near the Mission district.

\[\text{inform(food=cantonese, area=mission)}.\]

Dialogue act interpretation algorithm:

- 1-of-N supervised classification to choose inform
- Based on encodings of current sentence + prior dialogue acts

Simple dialogue state tracker:

- Run a slot-filler after each sentence
An special case of dialogue act detection: Detecting Correction Acts

If system misrecognizes an utterance
User might make a correction
  ◦ Repeat themselves
  ◦ Rephrasing
  ◦ Saying “no” to a confirmation question
Corrections are harder to recognize!

- From speech, corrections are misrecognized twice as often (in terms of word error rate) as non-corrections! (Swerts et al 2000)
- Hyperarticulation (exaggerated prosody) is a large factor:

"I said BAL-TI-MORE, not Boston"
Features for detecting corrections in spoken dialogue

<table>
<thead>
<tr>
<th>features</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexical</td>
<td>words like “no”, “correction”, “I don’t”, swear words, utterance length</td>
</tr>
<tr>
<td>semantic</td>
<td>similarity (word overlap or embedding dot product) between the candidate</td>
</tr>
<tr>
<td></td>
<td>correction act and the user’s prior utterance</td>
</tr>
<tr>
<td>phonetic</td>
<td>phonetic overlap between the candidate correction act and the user’s</td>
</tr>
<tr>
<td></td>
<td>prior utterance (i.e. “WhatsApp” may be incorrectly recognized as “What’s</td>
</tr>
<tr>
<td></td>
<td>up”)</td>
</tr>
<tr>
<td>prosodic</td>
<td>hyperarticulation, increases in F0 range, pause duration, and word</td>
</tr>
<tr>
<td></td>
<td>duration, generally normalized by the values for previous sentences</td>
</tr>
<tr>
<td>ASR</td>
<td>ASR confidence, language model probability</td>
</tr>
</tbody>
</table>
Chatbots and Dialogue Systems

The Dialogue-State Architecture
The Dialogue-State Architecture Continued: Policy and Generation
Dialogue Policy

At turn $i$ predict action $A_i$ to take, given entire history:

$$\hat{A}_i = \arg\max_{A_i \in A} P(A_i | (A_1, U_1, ..., A_{i-1}, U_{i-1}))$$

Simplify by just conditioning on the current dialogue state (filled frame slots) and the last turn and turn by system and user:

$$\hat{A}_i = \arg\max_{A_i \in A} P(A_i | \text{Frame}_{i-1}, A_{i-1}, U_{i-1})$$
Policy example: Confirmation and Rejection

Dialogue systems make errors
So they to make sure they have understood user
Two important mechanisms:
• confirming understandings with the user
• rejecting utterances that the system is likely to have misunderstood.
Explicit confirmation strategy

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 strategy

Explicit confirmation strategy:

U: Which city do you want to leave from?
S: Baltimore.
U: Do you want to leave from Baltimore?
S: 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 strategy:

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?
Confirmation strategy tradeoffs

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 also awkward and increases the length of the conversation (Danieli and Gerbino 1995, Walker et al. 1998).
Rejection

I’m sorry, I didn’t understand that.
Progressive prompting for rejection

Don't just repeat the question "When would you like to leave?"
Give user guidance about what they can say:

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.
Using confidence to decide whether to confirm:

ASR or NLU systems can assign a confidence value, indicating how likely they are that they understood the user.

- Acoustic log-likelihood of the utterance
- Prosodic features
- Ratio of score of best to second-best interpretation

Systems could use set confidence thresholds:

\begin{align*}
\text{\textless} \alpha & \quad \text{low confidence} \quad \text{reject} \\
\geq \alpha & \quad \text{above the threshold} \quad \text{confirm explicitly} \\
\geq \beta & \quad \text{high confidence} \quad \text{confirm implicitly} \\
\geq \gamma & \quad \text{very high confidence} \quad \text{don’t confirm at all}
\end{align*}
Natural Language Generation

NLG in information-state architecture modeled in two stages:

• **content planning** (what to say)
• **sentence realization** (how to say it).

We'll focus on sentence realization here.
Sentence Realization

Assume content planning has been done by the dialogue policy

- Chosen the dialogue act to generate
- 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
2 samples of Input and Output for Sentence Realizer

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>recommend(restaurant name= Au Midi, neighborhood = midtown, cuisine = french)</code></td>
<td>1) Au Midi is in Midtown and serves French food. 2) There is a French restaurant in Midtown called Au Midi.</td>
</tr>
<tr>
<td><code>recommend(restaurant name= Loch Fyne, neighborhood = city centre, cuisine = seafood)</code></td>
<td>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.</td>
</tr>
</tbody>
</table>
Sentence Realization

Training data is hard to come by

- Don't see each restaurant in each situation

Common way to improve generalization:

- **Delexicalization**: replacing words in the training set that represent slot values with a generic placeholder token:

```
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.
Sentence Realization

Training data is hard to come by
- Don't see each restaurant in each situation

Common way to improve generalization:
- **Delexicalization**: replacing words in the training set that represent slot values with a generic placeholder token:

```
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**.
Sentence Realization: mapping from frames to delexicalized sentences

Encoder-decoder models:

Output:

restaurant_name has decent service

Relexicalize to:

Au Midi has decent service
Generating clarification questions

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

The system repeats “going” and “on the 5th” to make it clear which aspect of the user’s turn the system needs to be clarified.

Methods for generating clarification questions:
• Rules like 'replace “going to UNKNOWN WORD” with “going where”'
• Classifiers that guess which slots were misrecognized
The Dialogue-State Architecture Continued: Policy and Generation
Evaluating Dialogue Systems
Evaluating chatbots and task-based dialogue

Task-based dialogue:
• mainly by measuring task performance

Chatbots:
• mainly by human evaluation
Chatbots are evaluated by humans

**Participant evaluation**: The human who talked to the chatbot assigns a score

**Observer evaluation**: third party who reads a transcript of a human/chatbot conversation assigns a score.
Participant evaluation

Human chats with model for 6 turns and rates 8 dimensions of quality:

- avoiding repetition, interestingness, making sense, fluency, listening, inquisitiveness, humanness, engagingness,

(1) Avoiding Repetition: How repetitive was this user?
  - Repeated themselves over and over • Sometimes said the same thing twice • Always said something new

(3) Making sense: How often did this user say something which didn't make sense?
  - Never made any sense • Most responses didn’t make sense • Some responses didn’t make sense • Everything made perfect sense

(8) Engagingness: How much did you enjoy talking to this user?
  - Not at all • A little • Somewhat • A lot

Observer evaluation: acute-eval


Annotators look at two conversations (A + B) and decide which is better:

**Engagingness:** Who would you prefer to talk to for a long conversation?

**Interestingness:** If you had to say one of these speakers is interesting and one is boring, who would you say is more interesting?

**Humanness:** Which speaker sounds more human?

**Knowledgeable:** If you had to say that one speaker is more knowledgeable and one is more ignorant, who is more knowledgeable?
The ACUTE-EVAL method
Li et al., 2019

Figure 1: The ACUTE-EVAL method asks humans to compare two multi-turn dialogues, and independent of the gray speakers, choose between Speaker 1 (light blue) and Speaker 2 (dark blue).

Automatic evaluation is an open problem

Automatic evaluation methods (like the BLEU scores used for Machine Translation) are generally not used for chatbots.

• They correlate poorly with human judgements.

One current research direction: **Adversarial Evaluation**

• Inspired by the Turing Test
• train a "Turing-like" classifier to distinguish between human responses and machine responses.
• The more successful a dialogue system is at fooling the evaluator, the better the system.
Task-based systems are evaluated by task success!

1. End-to-end evaluation (Task Success)
2. Slot Error Rate for a Sentence
   $\frac{\text{# of inserted/deleted/substituted slots}}{\text{# of total reference slots for sentence}}$
Evaluation Metrics: Slot error rate

“Make an appointment with Chris at 10:30 in Gates 104”

<table>
<thead>
<tr>
<th>Slot</th>
<th>Filler</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON</td>
<td>Chris</td>
</tr>
<tr>
<td>TIME</td>
<td>11:30 a.m.</td>
</tr>
<tr>
<td>ROOM</td>
<td>Gates 104</td>
</tr>
</tbody>
</table>

Slot error rate: 1/3

Task success: At end, was the correct meeting added to the calendar?
More fine-grained metrics: User Satisfaction Survey


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

**Efficiency cost:**
- total elapsed time for the dialogue in seconds,
- the number of total turns or of system turns
- total number of queries
- “turn correction ratio”: % of turns that were used to correct errors

**Quality cost:**
- number of ASR rejection prompts.
- number of times the user had to barge in
Chatbots and Dialogue Systems

Evaluating Dialogue Systems
Design and Ethical Issues
1. Study the users and task
   • value-sensitive design
2. Build simulations
   • Wizard of Oz study
3. Iteratively test design on users


Ethical design

Ethical issues have long been known to be crucial in artificial agents

Mary Shelley's *Frankenstein*

- creating agents without a consideration of ethical and humanistic concerns

Ethical issues:

- **Safety**: Systems abusing users, distracting drivers, or giving bad medical advice
- **Representational harm**: Systems demeaning particular social groups
- **Privacy**: Information Leakage
Safety

Chatbots for mental health
  ◦ Extremely important not to say the wrong thing

In-vehicle conversational agents
  ◦ Must be aware of environment, driver's level of attention

Abuse and Representation Harm: The case of Microsoft Tay

Experimental Twitter chatbot launched in 2016
- given the profile personality of an 18- to 24-year-old American woman
- could share horoscopes, tell jokes,
- asked people to send selfies
- used informal language, slang, emojis, and GIFs,
- Designed to learn from users (IR-based)
The case of Microsoft Tay

Immediately Tay turned offensive and abusive
- Obscene and inflammatory tweets
- Nazi propaganda, conspiracy theories
- Began harassing women online
- Reflecting racism and misogyny of Twitter users

Microsoft took Tay down after 16 hours

Lessons:
- User response must be considered in the design phase

Bias in training datasets

Henderson et al. ran hate-speech and bias detectors on standard training sets for dialogue systems:
- Twitter, Reddit, other dialogue datasets

Found bias and hate-speech
- In training data
- In dialogue models trained on the data

Privacy: Training on user data

Accidental information leakage

• “Computer, turn on the lights [answers the phone] Hi, yes, my password is...”
• Henderson show in simulation that this leakage can occur.

Intentional information leakage

• Dialogue systems that are designed to send user data to developer or advertisers
• Important to consider privacy-preserving dialogue systems


Chatbots and Dialogue Systems

Design and Ethical Issues