



Language Understanding

09 - Conversational Agents

Hossein Zeinali



Introduction



- According to Chatbot Magazine, a **Conversational Agent** is:
“A conversational agent is a software program which interprets and responds to statements made by users in ordinary natural language. It integrates computational linguistics techniques with communication over the internet.”
- Also known as **Dialog Agent**
 - Phone-based Personal Assistants
 - SIRI, Cortana, Google Now, Alexa
 - Talking to your car
 - Communicating with robots
 - Clinical uses for mental health
 - Chatting for fun



Two Classes of Systems

- Goal-based (task-oriented) dialog agents
 - Designed for a particular task
 - Uses conversation with users to help complete tasks
 - SIRI, interfaces to cars, robots
- Chatbots
 - According to Oxford Dictionaries, a chatbot is:
“A computer program designed to simulate conversation with human users, especially over the Internet.”
 - Systems designed for extended conversations
 - Mimic the unstructured conversations or ‘chats’ characteristic of human-human interaction



Properties of Human Conversation

- Turns
 - A dialogue is a sequence of turns (A1, B1, A2, and so on), each a single contribution to the dialogue
 - Can consist of a sentence or multiple sentences
- A system has to know when to start talking and when to stop talking.
- Endpointing or endpoint detection:
 - A system must detect whether a user is done speaking
- Speech Acts
 - Each utterance in a dialogue is a kind of **action** being performed by the speaker.





Properties of Human Conversation



- Grounding
 - A dialogue is a collective act performed by the speaker and the hearer.
 - The participants establish what they both agree on, called the **common ground**.
 - Speakers do this by **grounding** each other's utterances.
 - It means acknowledging that the hearer has understood the speaker.
 - Principle of closure – agents performing an action require evidence that they have succeeded in performing it
- Sub-dialogues and Dialogue Structure:
 - **Adjacency pairs** are composed of a **first pair part** and a **second pair part**.
- Inference



Chatbots

- The simplest kinds of dialogue systems
 - Can carry on extended conversations
- These systems often have an entertainment value
- Chatbot architectures:
 - Rule-based systems:
 - E.g. ELIZA and PARRY systems
 - Corpus-based systems:
 - Mine large datasets of human-human conversations
 - Information retrieval based
 - IR-based systems simply copy a human's response from a previous conversation
 - Machine translation based
 - Learn to map from a user utterance to a system response



Rule-based Chatbots: ELIZA

- ELIZA was designed to simulate a Rogerian psychologist
 - By Joseph Weizenbaum, MIT, 1966
 - Draw the patient out by reflecting patient's statements back at them
- ELIZA worked by pattern/transform rules like:
 - (0 YOU 0 ME) [pattern]
->
(WHAT MAKES YOU THINK I 3 YOU) [transform]
You hate me
->
WHAT MAKES YOU THINK I HATE YOU
 - (I *) -> (You say you 2)
I know everybody laughed at me
YOU SAY YOU KNOW EVERYBODY LAUGHED AT YOU



Rule-based Chatbots: ELIZA

Welcome to

```
EEEEEE LL      IIII ZZZZZZZZ AAAAA
EE      LL      II      ZZ  AA  AA
EEEEEE LL      II      ZZZ  AAAAAAA
EE      LL      II      ZZ  AA  AA
EEEEEE LLLLLL IIII ZZZZZZZZ AA  AA
```

Eliza is a mock Rogerian psychotherapist.

The original program was described by Joseph Weizenbaum in 1966.

This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?

YOU: Men are all alike.

ELIZA: What is the connection, do you suppose ?

YOU: They're always bugging us about something or other.

ELIZA: Can you think of a specific example ?

YOU: Well, my boyfriend made me come here.

ELIZA: Is it important to you that your boyfriend made you come here ?

YOU: He says I'm depressed much of the time.

ELIZA: I am sorry to hear that you are depressed.

YOU: It's true. I am unhappy.

ELIZA: Can you explain what made you unhappy ?

YOU: █



Corpus-based Chatbots

- Corpus-based Chatbots, instead of using hand-built rules, mine conversations of human-human conversations
- These systems are enormously data-intensive
 - Require hundreds of millions or even billions of words.
- Most corpus-based chatbots do very little modeling of the conversational context.
 - Like rule-based chatbots but unlike frame-based dialogue systems
 - They tend to focus on generating a single response turn that is appropriate given the user's immediately previous utterance or two.
 - Similar to question answering systems



IR-Based Chatbots

- Response to a user's turn X by repeating some appropriate turn Y from a corpus of natural (human) text.
- Can use any retrieval algorithm to choose an appropriate response from the corpus.

- Return the response to the most similar turn

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

- Return the most similar turn

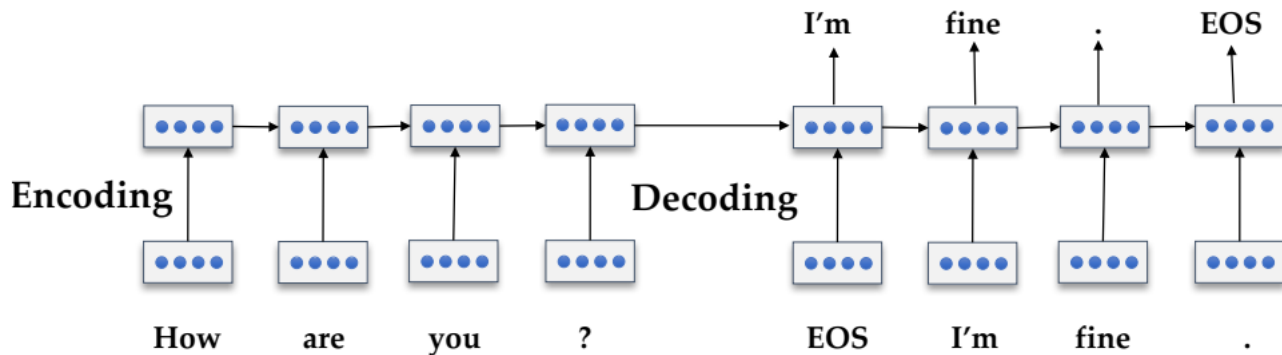
$$r = \operatorname{argmax}_{t \in C} \frac{q^T t}{\|q\| \|t\|}$$

- 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.



Encoder-decoder Chatbots

- Generate response as a task of transducing from the user's prior turn to the system's turn.
- Comparison with machine translation:
 - In MT, 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.



Encoder-decoder Chatbots

- Shortcoming of the basic encoder-decoder model:
 - Have a tendency to produce predictable but repetitive responses
 - Can be addressed by changing the objective function to a mutual information objective
 - Or by modifying a beam decoder to keep more diverse responses in the beam.
 - Inability to model the longer prior context of the conversation
 - Can be done by allowing the model to see prior turns, such as by using a hierarchical model
 - Focus on generating single responses
 - Can be addressed by using reinforcement learning, as well as techniques like adversarial networks



Encoder-decoder Chatbots

- Shortcoming of the basic encoder-decoder model:
 - Have a tendency to produce predictable but repetitive responses
 - Can be addressed by changing the objective function to a mutual information

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 ?



Evaluating Chatbots

- Chatbots are generally evaluated by humans.
 - It's a difficult task to evaluate it using machine
 - The slot-filling evaluations aren't appropriate for this task
 - Other metrics like BLEU correlate very poorly with human judgments
- Recently proposed evaluation methods:
 - Adversarial evaluation
 - Train a *Turing-like* evaluator classifier to distinguish between human-generated responses and machine-generated responses.
 - ADEM
 - An evaluation model that learns to predict human-like scores to input responses, using a new dataset of human response scores.



Types of dialogue systems

- **Command and control**
 - Actions in the world
 - Robot – situated interaction
- **Information access**
 - Database access
 - Bus/train/airline information
 - Librarian
 - Voice manipulation of a personal calendar
 - API access
- **IVRs – customer service**
 - Simple call routing
 - Menu-based interaction
 - Allows flexible response “How may I help you?”
- **Smart virtual assistant**
 - Helps you perform tasks, such as buying movie tickets, trouble shooting
 - Reminds you about important events without explicit reminder settings

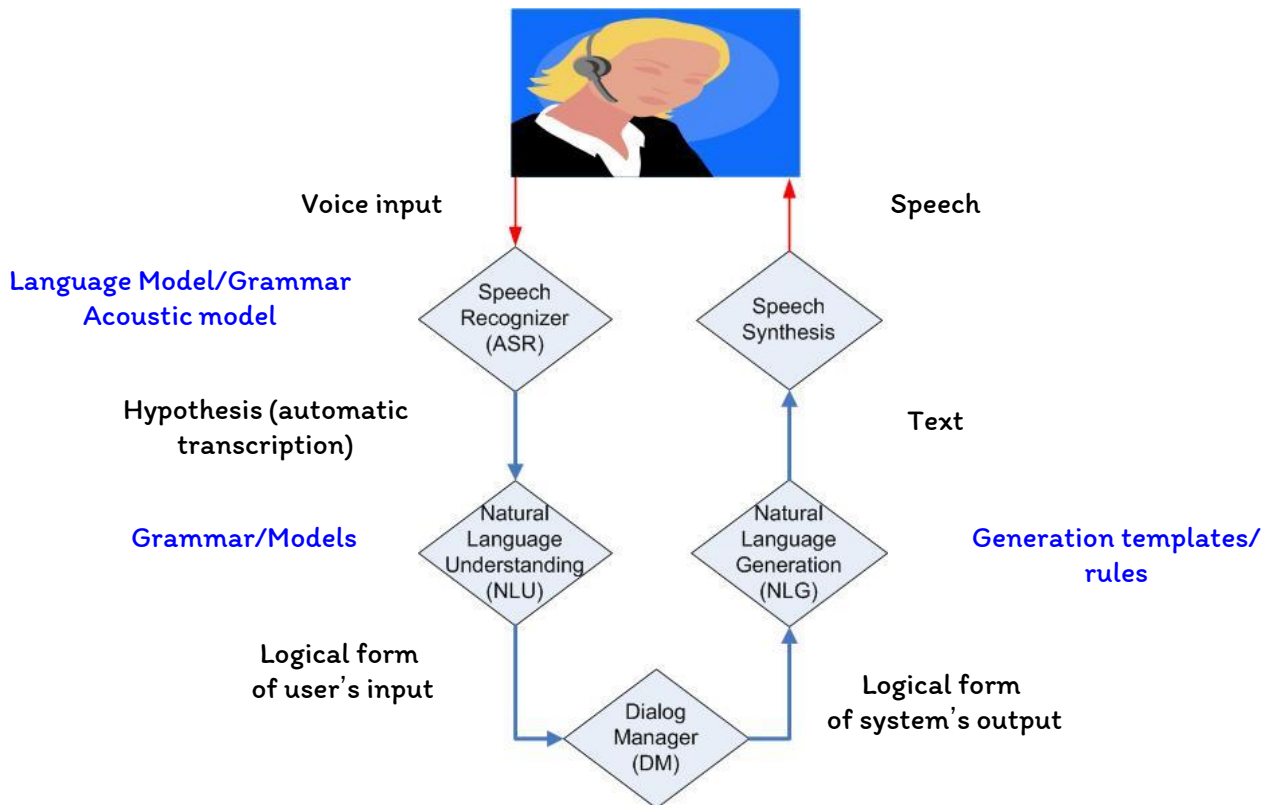


Aspects of Dialogue Systems

- Which modalities does the system use
 - Voice only (telephone/microphone & speaker)
 - Voice and graphics (smartphones)
 - Virtual human
 - Can show emotions
 - Physical device
 - Can perform actions
- Back-end
 - which resources (database/API/ontology) it accesses
- How much world knowledge does the system have
 - Hand-built ontologies
 - Automatically learned from the web
- How much personal knowledge does it have and use
 - Your calendar (Google)
 - Where you live/work (Google)
 - Who are your friends/relatives (Facebook)



Dialog System Components



Speech Recognition

- Convert speech signal into text
- Most SDS use off-the-shelf speech recognizers
 - Research systems are highly configurable:
 - Kaldi – most used research recognizer
 - Sphinx/pocket sphinx (java API)
 - Industry (free cloud version), not configurable
 - Google
 - Nuance
 - AT&T Watson



Speech Recognition

- Statistical process
- Use acoustic models that maps signal to phonemes
- Use language models (LM)/grammars that describe the expected language
- Domain types:
 - Open-domain speech recognition use LM built on large corpora
 - Specific-domain speech recognition
 - Adapting LMs to the domain



Speech Recognition

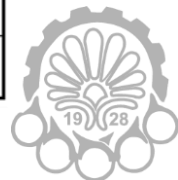
- Challenges: recognition errors due to

- Noisy environment
- Speaker accent
- Speaker interruption, self correction, etc.

- Speaker dependency

- Domain dependency

SYS:	Welcome to the CMU Let's Go bus information. What can I do for you?
USR:	I need to go from Oakland
ASR:	<i>I need to go .from. can't</i>
SYS:	Where do you wanna leave from?
USR:	to Waterfront
ASR:	<i>told. .me. Waterfront</i>
SYS:	Leaving from Waterfront . Did I get that right?
USR:	Oakland
ASR:	<i>Oakland</i>
	...



Speech Recognition

- Grammar-based
 - Allows dialogue designer to write grammars
 - For example, if your system expects digits, a rule is:
 - $S \rightarrow \text{zero} \mid \text{one} \mid \text{two} \mid \text{three} \mid \dots$
 - Advantages: better performance on in-domain speech
 - Disadvantages: does not recognize out-of-domain
- Open Domain – large vocabulary
 - Use language models built on large diverse dataset
 - Advantages: can potentially recognize any word sequence
 - Disadvantages: lower performance on in-domain utterances (digits may be misrecognized)



Natural Language Understanding

- Convert input text into internal representation.
Example internal representation in wit.ai:
- Can be based on simple phrase matching
 - “leaving from PLACE”
 - “arriving at TIME”
- Can use deep or shallow syntactic parsing
- Can be rule-based
 - Rules define how to extract semantics from a string/syntactic tree
- Or Statistical
 - Train statistical models on annotated data
 - Classify intent
 - Tag named entities

```
{  
  "msg_body": "what is playing at Lincoln  
Center",  
  "outcome": {  
    "intent": "get_shows",  
    "entities": {  
      "Venue": {  
        "value": "Lincoln Center",  
      }  
    },  
    "confidence": 0.545  
  },  
  "msg_id": "c942ad0f-0b63-415f-b1ef-  
84fbfa6268f2"  
}
```



Dialogue Manager (Policy)

- Is a “brain” of an SDS
- Decides on the next system action/dialogue contribution
 - At turn i in the conversation we want to predict which action A_i to take, based on the entire dialogue state.

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

- Can be simplified by maintaining as the dialogue state mainly just the set of slot-fillers that the user has expressed.
- This module has to determine whether sufficient information has been elicited from the user in order to do expected task.
 - Is also responsible for detecting and repairing breakdowns in the dialogue through verifications, confirmations, and corrections.



Dialogue Manager (Policy)

- Modern dialogue systems often make mistakes.
 - **Explicit confirmation:** system asks the user a direct question to confirm.
 - **Implicit confirmation:** system instead can demonstrate its understanding as a **grounding** strategy.
 - **Rejection:** system gives the user a prompt like “I’m sorry, I didn’t get that.”
- Dialog management systems can be categorized in terms of type of control offered and how the control is managed.
- Dialogues control may be:
 - System-led: the system asks a sequence of questions to fill the slots.
 - User-led: the user asks system questions to obtain the information.
 - Mixed initiative: the dialogue is shared.



Dialogue Manager (Policy)

- DM approaches:
 - Graph-based
 - Represent the dialogue manager as a graph or flow-chart
 - Sometimes called the call-flow or finite state based dialogue control
 - The nodes of graph represent system questions, the transitions between the nodes represent answers to questions, and graph specifies all legal dialogue.
 - Frame-based
 - The user is asked questions that enable the system to fill slots in a template in order to perform a task.
 - Statistical
 - Learn state transition rules from data or on-line and is able to deal with uncertainty
 - Modeling by Markov decision process or augment the hand crafted belief states
 - Hybrid
 - A combination of graph/frame based method with statistical method



Natural Language Generation Approaches

- Presenting semantic content to the user
- Template-based
 - E.g.: In a airline reservation system:
 - User: “Find me a ticket from **New York** to **London**”
 - System: “What date do you want to travel?”
 - User: “**March 10**”
 - System: “There is a **United** flight from **Newark airport** to **London Heathrow** on **March 10** leaving at **9:15 AM**”
 - Template: There is a **AIRLINE** flight from **AIRPORT** to **AIRPORT** on **DATE** leaving at **TIME**
 - Templates made by **delexicalization**
 - Generally done by encoder decoder models
- Is often modeled in two stages
 - Content planning (what to say)
 - Sentence realization (how to say it)
 - We will see data-to-text generation



Natural Language Generation (NLG)

- Content selection

- User asks “Find me restaurants in Chelsea”
- System finds 100 restaurants
- NLG decides how to present a response and which information to present
 - “I found 100 restaurants, the restaurant with highest rating is ...”
 - “I found 100 restaurants, the closest to you is ...”
 - “I found 100 restaurants, I think you would like ...”



How SIRI works

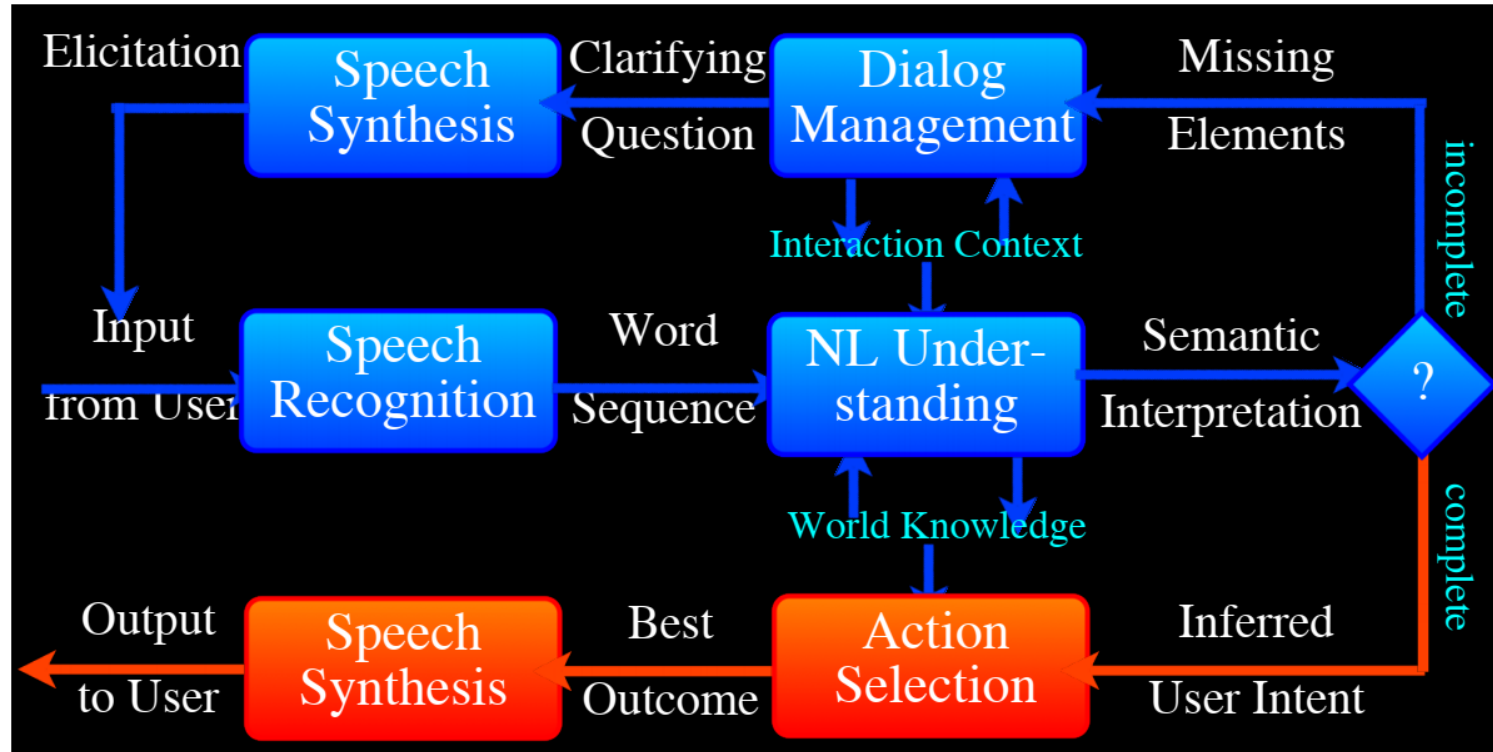
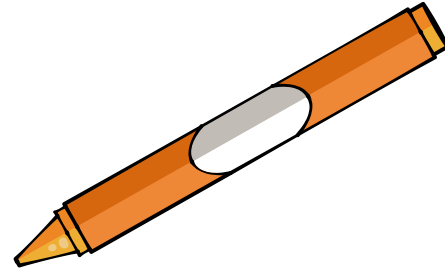
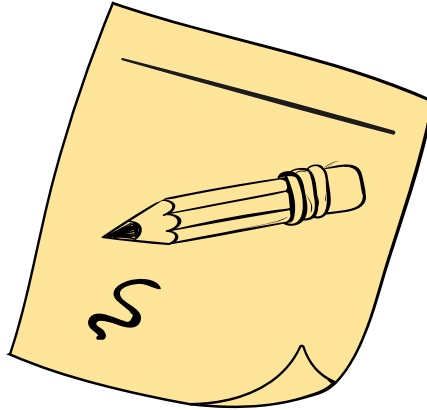


Figure from Jerome Bellegarda



Frame-based Dialogue Systems



GUS: Simple Frame-based Dialogue Systems

- Architecture introduced in GUS travel planning system (Bobrow et al., 1977)
 - Is a kind of task-based dialogue
- This architecture is used in most modern commercial digital assistants
- Based on a domain ontology, which defines frames (collections of slots) and values that each frame's slot can take



An Actual Dialogue with the GUS System

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



GUS: Simple Frame-based Dialogue Systems

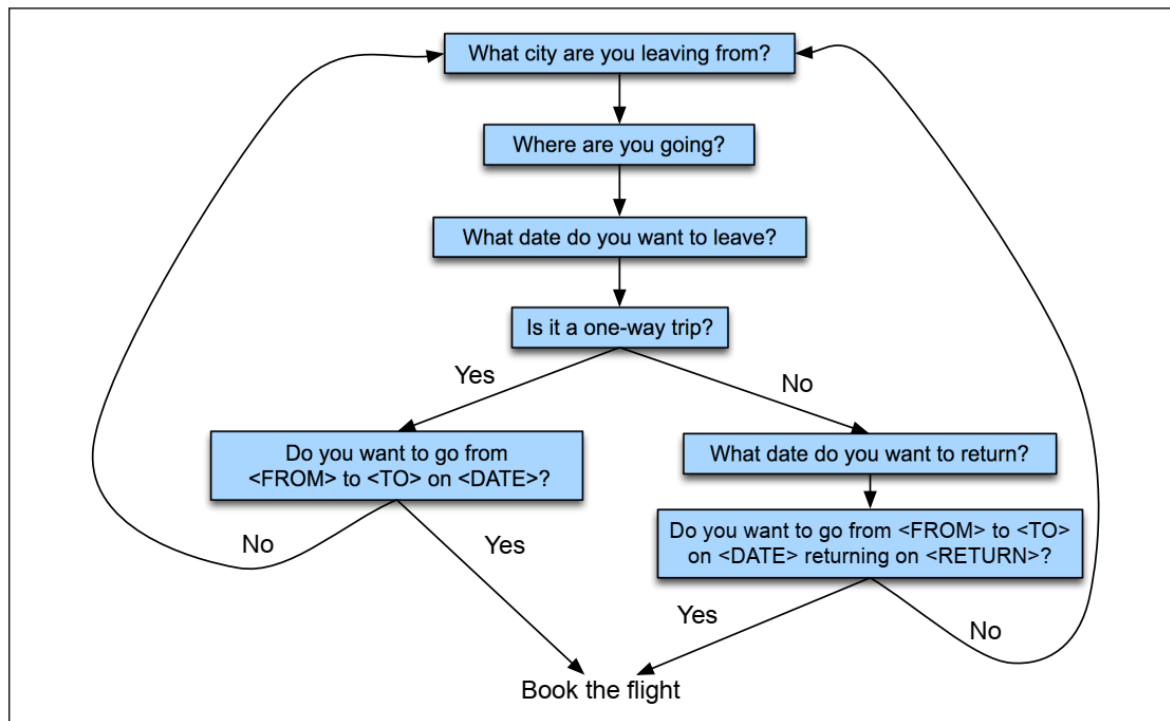
- The set of slots specifies what the system needs to know
- The filler of each slot is constrained to values of a particular semantic type

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?”

- Types in GUS have hierarchical structure; for example the *date* type



Control Structure for Frame-based Dialogue



A simple finite-state automaton architecture for frame-based dialog.



GUS Architecture

- GUS architecture is **production rule** system
 - A computer program typically used to provide some form of artificial intelligence.
 - A set of rules about behavior and also includes the mechanism necessary to follow those rules.
- Has a dynamic, flexible and mixed system
 - Can fill slots in any order
 - Can be used in both system and user lead
- Condition-action rules attached to slots

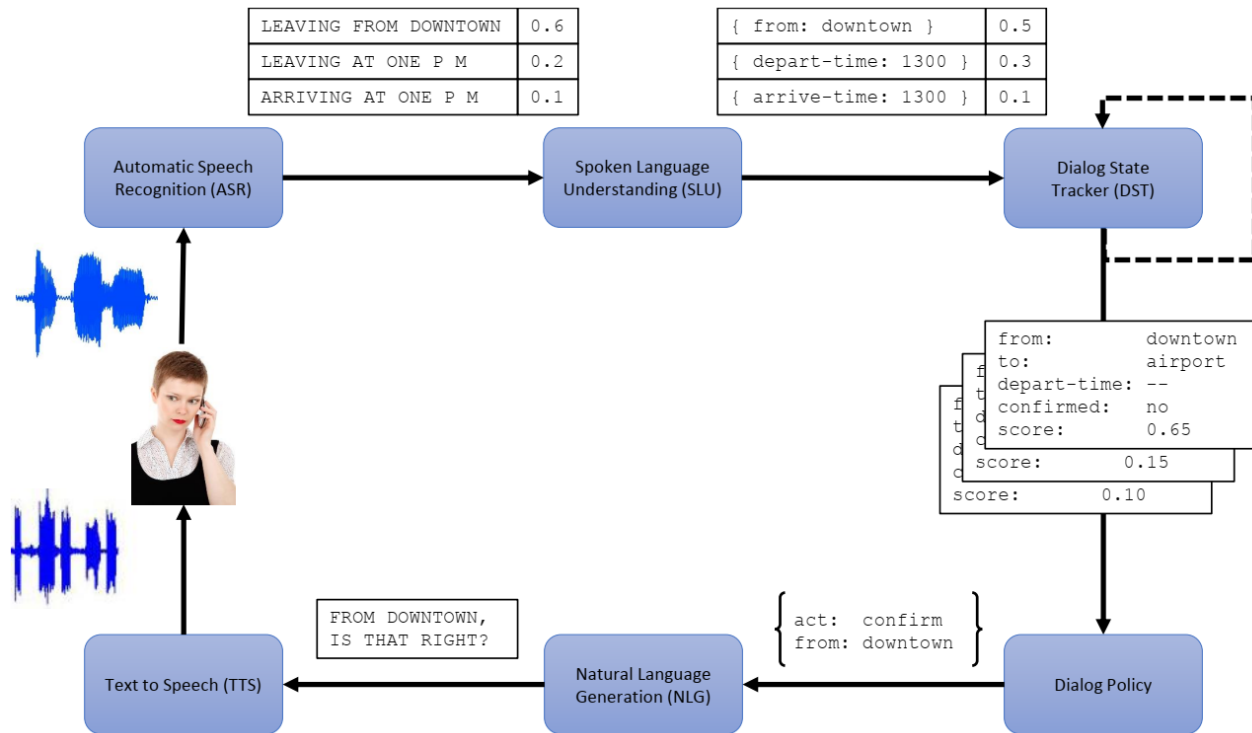


Dialogue-state Architecture

- It is used in modern task-based dialogue systems
- Capable of understanding and generating **dialogue acts**
 - **Dialogue acts:** utterance in the context of a dialogue that serves a function in the dialogue
 - E.g. question, statement, request for action, acknowledgment
- Dialogue policy – decide what to do or say
- Dialogue state tracker – maintain the current state of the dialogue



Dialogue-state Architecture



Dialogue Acts

- A **dialog act** is an utterance that serves a function in the dialog.
 - Ask questions, make promises, pay compliments, and so on.

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



Dialogue State Tracking

- Determine the current state of the frame (the fillers of each slot), as well as the user's most recent dialogue act.
- Dialogue acts place some constraints on the slots and values
 - The tasks of dialogue-act (intent) detection and slot-filling are often performed jointly.

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



Evaluating Dialogue Systems

- Task success
 - E.g. did the system book the correct flight?
- Performance of SDS components
 - ASR (WER)
 - NLU (slot error rate)
 - DM/NLG (appropriate response)
- Efficiency cost
 - The total elapsed time, the number of total turns, or the total number of queries, the number of system non-responses and the “turn correction ratio”



Evaluating Dialogue Systems

- Quality cost
 - Measures other aspects that affect users' perception
- Dialogue quality
 - Timeouts (when a user did not respond)
 - Rejects (when the system confidence is low leading to "I am sorry I did not understand")
 - Help – number of times the system believes that a user said 'help'
 - Cancel - number of times the system believes that a user said 'cancel'



Evaluating Dialogue Systems

- User satisfaction rating

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?



Furhat Robot





Thanks for your attention



References and IP Notice

- Daniel Jurafsky and James H. Martin, “Speech and Language Processing”, 3rd ed., 2019
- Some of the slides were selected from Sarah Ita Levitan's slides.
- Some graphics were selected from [Slidesgo](#) template

