Lecture 4: Dialog system design. GUS and frame-based dialog systems. Alexa Skills Kit.
Outline

- Dialog system design
- GUS and frame-based systems
- NLU and NLG considerations
- Alexa skills kit overview

Homework 1 due Monday 11:59pm
Extra Office Hours Monday
Spoken Dialog Agent Conceptual Architecture
Dialog System Design: User-centered Design

1. Study the user and task
2. Build simulations "Wizard of Oz study"
3. Iteratively test the design on users
4. Build a system to meet most valuable (and feasible) needs

Gould and Lewis 1985
System design considerations

- Goal and scope of overall system?
  - What tasks/actions are supported?
  - What state do the dialog/task managers track?
- What level of interaction complexity?
  - Initiative? Back-tracking? NLU support for paraphrasing?
- Need a solution for each module (ASR, TTS, NLU, NLG, task/dialog manager)
- What is the interface / data structure between modules?
  - e.g. Does ASR module send transcripts only? Emotion labels? Audio?
Case study: GoButler

- Text chat interface
- *Human* operator could complete any task!
  - Canceling cable subscriptions, booking restaurants etc.
- Wave of startups 2013-2017
  - Magic, Operator, Facebook M
- Idea: Gather enough data to automate most tasks
- With enough data, NLP + connected services would allow positive unit economics
Case study: GoButler

Current product (2022)

What happened to *do anything for me*?

---

**Your Personal Data Assistant**

GoButler crawls the internet and public records and remembers all the contact information he comes across.

He does this so he can be the best possible assistant for you. Whoever you need to find or get a hold of, GoButler is ready to serve you.

---

**Secure Free People Search**

GoButler provides an incredible amount of information for free, and aims to be dead simple to work with.

If you want additional information, GoButler can refer you to others he trusts to get you what you need.

---

GoButler, your assistant who knows everyone

GoButler can tell you the contact information for nearly anyone. Phone numbers, email addresses, social media profiles and more.

<table>
<thead>
<tr>
<th>Phone</th>
<th>Email</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>415-123-1234</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

no credit card required
Case study: GoButler

- The “personal assistants to do anything” startup hype wave settled down 2017+
- Collecting data isn’t enough to cover *all* possible tasks
  - Acting on the task often complex. Requires a human.
  - NLP/Dialog aspects work fairly well for simpler requests
- What came out of these experiments?
  - Narrow-domain chat assistants for valuable services
  - Tools for quickly designing and building dialog assistants
  - Lots of VC money spent on users getting free personal assistants for a while 😃

[TechCrunch article on GoButler pivot](#)
Rough system design process: Design phase

1. Overall system goal.
2. Define set of task actions system can perform
3. Create example interactions
Rough system design process: Technology choices after design

1. Overall system goal.
2. Define set of task actions system can perform
3. Create example interactions
4. Define dialog manager approach (actions + dialog acts/state of system)
5. Choose NLU approach matching complexity of tasks and approach to initiative + dialog acts
6. Define NLG approach and dialog state -> NLG interface
7. Create a dialog policy (choosing next dialog action and sending to NLG)
8. Choose ASR/TTS approach. Update NLU/NLG if needed
System design considerations

- Not all systems require support for complex interactions
- Frameworks like Alexa force some choices about multiple modules to simplify overall development
- ASR/TTS components often be treated as black-box, but great systems are sensitive to ASR uncertainty
- Okay to redefine/combine modules based on problem (e.g. a smart NLG module might simplify dialog manager)
Outline

- Dialog system design
- **GUS and frame-based systems**
- NLU and NLG considerations
- Alexa skills kit overview
Frame-based dialog agents

- One or more **frames**
  - Each a collection of **slots**
  - Each slot having a **value**
- **Domain ontology** is full collection of frames for a (sub)system
- A frame defines the task data structure
The Frame

- A set of **slots**, to be filled with information of a given **slot 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>What city are you leaving from?</td>
</tr>
<tr>
<td>DEST</td>
<td>city</td>
<td>Where are you going?</td>
</tr>
<tr>
<td>DEP DATE</td>
<td>date</td>
<td>What day would you like to leave?</td>
</tr>
<tr>
<td>DEP TIME</td>
<td>time</td>
<td>What time would you like to leave?</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>line</td>
<td>What is your preferred airline?</td>
</tr>
</tbody>
</table>
Frame-based dialogue agents

1977:

GUS, A Frame-Driven Dialog System

Daniel G. Bobrow, Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson and Terry Winograd

Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, U.S.A.

Artificial Intelligence Journal, 1977

- Still the industrial state of the art
- Alexa roughly follows GUS architecture
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 8.30 am and arrives at 10.00 am
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.
Slot types can be complex, hierarchical

- The type *DATE*

```plaintext
DATE
  MONTH NAME
  DAY (BOUNDDED-INTEGER 1 31)
  YEAR INTEGER
  WEEKDAY (MEMBER (SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY))
```
Simple action policy for frame-based dialog

Consider a trivial airline travel system:
Ask the user for a departure city
Ask for a destination city
Ask for a time
Ask whether the trip is round-trip or not
Finite State Dialog Manager

What city are you leaving from?

Where are you going?

What date do you want to leave?

Is it a one-way trip?

Yes

Do you want to go from <FROM> to <TO> on <DATE>?

No

What date do you want to return?

Yes

Do you want to go from <FROM> to <TO> on <DATE> returning on <RETURN>?

No

Book the flight
Finite-state dialog managers

- System completely controls the conversation with the user.
- It asks the user a series of questions.
- Ignoring (or misinterpreting) anything the user says that is not a direct answer to the system’s questions.
Frames and mixed initiative

- System asks questions of user, filling any slots that user specifies
  - When frame is filled, do database query
- If user answers 3 questions at once, system can fill 3 slots and not ask these questions again!
- Frame structure guides dialog
Mixed Initiative

- Conversational initiative can shift between system and user
- Simplest kind of mixed initiative: use the structure of the frame to guide dialogue

<table>
<thead>
<tr>
<th>Slot</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIGIN</td>
<td>What city are you leaving from?</td>
</tr>
<tr>
<td>DEST</td>
<td>Where are you going?</td>
</tr>
<tr>
<td>DEPT DATE</td>
<td>What day would you like to leave?</td>
</tr>
<tr>
<td>DEPT TIME</td>
<td>What time would you like to leave?</td>
</tr>
<tr>
<td>AIRLINE</td>
<td>What is your preferred airline?</td>
</tr>
</tbody>
</table>
NLU and NLG with frame-based systems
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
Natural Language Understanding for filling slots

Wake me tomorrow at six.

DOMAIN: ALARM-CLOCK
INTENT: SET-ALARM
TIME: 2017-07-01 0600-0800
Rule-based Slot-filling

Write regular expressions or grammar rules

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

Do text normalization

Time consuming and brittle NLU capabilities

*With modern NLP tools/features, only use rules alone in special cases*
Machine learning for slot-filling

I want to fly to San Francisco on Monday afternoon please

Use 1-of-N classifier for Domain/Intent. Use sequence model to tag words/phrases with slot names

- **Input:**
  - features like word N-grams

- **Output:**
  - Domain: AIRLINE
  - Intent: SHOWFLIGHT
  - Destination-City: “San Francisco”
  - Depart-Date: “Monday”
More sophisticated algorithm for slot filling: IOB Tagging

- IOB Tagging
  - tag for the beginning (B) and inside (I) of each slot label,
  - plus one for tokens outside (O) any slot label.
  - $2n + 1$ tags, where $n$ is the number of slots.

B-DESTINATION
I-DESTINATION
B-DEPART_TIME
I-DEPART_TIME
O

```
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
```
Slot-filling is information/entity extraction from text NLP

- Conditional Random Field (CRF) with word vector features, or neural classifiers both work well

Example NER with CRF overview:

Back in 2000, People Magazine highlighted Prince William's style who at the time was a little more fashion-conscious, even making fashion statements at times.

Nowadays the prince mainly wears navy suits (sometimes double-breasted), light blue button-ups with classic pointed collars, and burgundy ties.

But who knows what the future holds...

Duchess Kate did wear an Alexander McQueen dress to the wedding in the fall of 2017.
Sequence models for slot filling: IOB Tagging

- IOB Tagging is done by a sequence model
- Typical:

Extracted strings can then be normalized (San Fran->SFO)
Generation Component (NLG)

- **Content Planner**
  - Decides what content to express to user
    - (ask a question, present an answer, etc)
  - Often merged with dialogue manager

- **Language Generation**
  - Chooses syntax and words

- **TTS**

- **In practice**: Template-based w/most words prespecified

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

- Dialogue manager builds representation of meaning of utterance to be expressed
- Passes this to a “generator”
- Mixing chatbot-like NLG constrained to convey dialog representation can improve user satisfaction
NLG in Dialog

• Great area to leverage pre-trained large language models / text generation
• Critical aspect: Ensure correctness of what we convey to the user!
NLG in Dialog

• Great area to leverage pre-trained large language models / text generation
• Critical aspect: Ensure correctness of what we convey to the user!
Deep learning NLG conditioned on dialog semantics

Figure 2: Illustration of SC-GPT. In this example, SC-GPT generates a new word token (e.g., “confirm” or “center”) by attending the entire dialog act and word tokens on the left within the response.
Semantically conditioned GPT for dialog NLG

\[ \mathcal{A} = [\mathbf{I}, (s_1, v_1), \ldots, (s_P, v_P)] \]

\[
p_\theta(x|\mathcal{A}) = \prod_{t=1}^{T} p_\theta(x_t|x_{<t}, \mathcal{A})
\]

Figure 2: Illustration of SC-GPT. In this example, SC-GPT generates a new word token (e.g., “confirm” or “center”) by attending the entire dialog act and word tokens on the left within the response.

(Peng et al., 2020)
Deep learning NLG conditioned on dialog semantics

<table>
<thead>
<tr>
<th>3</th>
<th><strong>Input</strong></th>
<th><strong>DA</strong></th>
<th><strong>Reference</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Input</strong></td>
<td>Train{inform(time=50 minutes)}, hotel{request(stars=?, area=?), inform(choice=5)}</td>
<td>travel time is 50 minutes. i found 5 hotels you may like. do you have a star rating request or an area you prefer</td>
</tr>
<tr>
<td></td>
<td><strong>Reference</strong></td>
<td>there are 5 trains that are 50 minutes. do you have a preferred departure location?</td>
<td>there are 5 trains that meet your criteria. do you have a preference on the area or star rating?</td>
</tr>
<tr>
<td></td>
<td><strong>SC-LSTM</strong></td>
<td><img src="image" alt="SC-LSTM example" /></td>
<td><img src="image" alt="SC-LSTM example" /></td>
</tr>
<tr>
<td></td>
<td><strong>HDSA</strong></td>
<td><img src="image" alt="HDSA example" /></td>
<td><img src="image" alt="HDSA example" /></td>
</tr>
<tr>
<td></td>
<td><strong>SC-GPT</strong></td>
<td><img src="image" alt="SC-GPT example" /></td>
<td><img src="image" alt="SC-GPT example" /></td>
</tr>
</tbody>
</table>

Table 8: Examples of generated utterances from different models, along with its corresponding dialog acts (DAs) and references. The first two examples are sampled from FEWSHOTWOZ and the last one is from MultiWOZ. Each generated utterance is followed by a brief description explaining the errors (starting with “%”). (Better viewed in color: wrong, redundant, missing information)

(Peng et al., 2020)
### Deep learning NLG conditioned on dialog semantics

<table>
<thead>
<tr>
<th></th>
<th><strong>Input DA</strong></th>
<th><strong>Reference</strong></th>
<th><strong>SC-LSTM</strong></th>
<th><strong>GPT-2</strong></th>
<th><strong>SC-GPT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Laptop{inform(name=satellite proteus 84; type=laptop; memory=8 gb; drive=1 tb; weight=2.3 kg)}</td>
<td>the satellite proteus 84 is a laptop with a 1 tb drive, 8 gb memory and weighs in at 2.3 kg</td>
<td>for business computing, and is used for business computing</td>
<td>[businesscomputing=true] [weight=2.3kg]</td>
<td>[memory=8 gb]</td>
</tr>
<tr>
<td></td>
<td><strong>Reference</strong></td>
<td>the satellite proteus 84 is a laptop with 8 gb, with a 1 tb drive, and is for business computing</td>
<td>[businesscomputing=true]</td>
<td>[weight=2.3kg]</td>
<td>[memory=8 gb]</td>
</tr>
<tr>
<td>2</td>
<td>Restaurant{inform_only_match(name=marnee thai and thanh long; pricerange=moderate; near=outer sunset; goodformeal=dinner)}</td>
<td>marnee thai and thanh long are the only 2 moderately priced restaurants near outer sunset that are good for dinner</td>
<td>marnee thai and thanh long is is moderate restaurant restaurant and good good good dinner</td>
<td>[near=outer sunset] [inform_only_match]</td>
<td>[inform_only_match]</td>
</tr>
<tr>
<td></td>
<td><strong>Reference</strong></td>
<td>marnee thai and thanh long are the only 2 moderately priced restaurants near outer sunset that are good for dinner</td>
<td>there is a moderately priced restaurant called marnee thai and thanh long that is near the outer sunset area</td>
<td>% [goodformeal=dinner] [inform_only_match]</td>
<td>[inform_only_match]</td>
</tr>
<tr>
<td></td>
<td><strong>SC-GPT</strong></td>
<td>marnee thai and thanh long is the only restaurant that serves moderately priced food near outer sunset and good for dinner</td>
<td>marnee thai and thanh long is the only restaurant that serves moderately priced food near outer sunset and good for dinner</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*(Peng et al., 2020)*
Outline

- Dialog system design
- GUS and frame-based systems
- NLU and NLG considerations
- Alexa skills kit overview
Alexa Skills Kit

- A *Skill* is a top level command for Alexa.
  - “Alexa open 224S Homework 2”
  - Skill → **domain ontology**
- A skill contains *intents* which are distinct task actions.
  - Intent → **frame**
  - Design intents with built-in capabilities per intent and ASK interaction model in mind
- Each intent contains *slots* which each have a *slot type* and take on a *slot value*
- Not quite this simple (e.g. ASK **built-in intents** are not simple to define in the frame/slot abstraction)
Alexa Skills Kit

- Dialog management is complex, partially handled with built-in features (clarification, value verification, cancel skill, etc)
- NLU through grammars and examples.
  - ASK trains models for you based on examples
  - Many rich slot types (dates, numbers, lists)
- Task management is custom! ASK provides a dialogue API to your web server, you implement server-side task execution.
- NLG is template-based with ASK adding variety
- ASR/TTS handled by ASK. Interface is text/transcripts
- Overall framework is API/SDK oriented like web dev
Alexa domain classification

Figure 1: The overall architecture of the personalized dynamic domain classifier.

(Kim et al, 2018)
# ASK interaction schema

## Interaction Model

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>languageModel</td>
<td>object</td>
<td>Conversational primitives for the skill</td>
<td>yes</td>
</tr>
<tr>
<td>dialog</td>
<td>object</td>
<td>Rules for conducting a multi-turn dialog with the user</td>
<td>no</td>
</tr>
<tr>
<td>prompts</td>
<td>array</td>
<td>Cues to the user on behalf of the skill for eliciting data or providing feedback</td>
<td>no</td>
</tr>
</tbody>
</table>

### languageModel

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>invocationName</td>
<td>string</td>
<td>Invocation name of the skill</td>
<td>yes</td>
</tr>
<tr>
<td>intents</td>
<td>array</td>
<td>Intents and their slots</td>
<td>yes</td>
</tr>
<tr>
<td>types</td>
<td>array</td>
<td>Custom slot types</td>
<td>no</td>
</tr>
<tr>
<td>modelConfiguration</td>
<td>object</td>
<td>Optional settings for the interaction model. Available in supported locales.</td>
<td>no</td>
</tr>
</tbody>
</table>

### languageModel_intents

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the intent. For details about intent names, see intent and slot name requirements.</td>
<td>yes</td>
</tr>
<tr>
<td>slots</td>
<td>array</td>
<td>List of slots within the intent.</td>
<td>no</td>
</tr>
<tr>
<td>samples</td>
<td>array</td>
<td>Sample utterances for the intent.</td>
<td>no</td>
</tr>
</tbody>
</table>
ASK Intent JSON example

This example shows a portion of the intent object for a PlanMyTrip intent. The utterances for the intent are in interactionModel.languageModel.intents[].samples. Each slot has its own samples array. For brevity, other properties within interactionModel and languageModel are not shown.

```json
{
  "interactionModel": {
    "languageModel": {
      "intents": [
        {
          "name": "PlanMyTrip",
          "slots": [
            {
              "name": "travelDate",
              "type": "AMAZON.DATE",
              "samples": ["I am taking this trip on {travelDate}", "on {travelDate}", "{travelDate}" ]
            },
            {
              "name": "toCity",
              "type": "AMAZON.US_CITY",
              "samples": ["I'm going to {toCity}", "{toCity}" ]
            },
            {
              "name": "fromCity",
              "type": "AMAZON.US_CITY",
              "samples": ["I'm starting from {fromCity}" ]
            },
            {
              "name": "travelMode",
              "type": "LIST_OF_TRAVEL_MODES",
              "samples": ["I am going to {travelMode}", 
                          "{travelMode}" ]
            },
            {
              "name": "activity",
              "type": "LIST_OF_ACTIVITIES",
              "samples": ["{activity}", "I plan to {activity}" ]
            }
          ],
          "samples": ["{toCity}", "I want to travel from {fromCity} to {toCity} {travelDate}", "i want to visit {toCity}", "i am going on trip on {travelDate}", "I'm {travelMode} from {fromCity} to {toCity}", "i'm {travelMode} to {toCity} to {activity}", "plan a trip", "plan a trip to {toCity}", "plan a trip starting from {fromCity}", "I'd like to leave on {travelDate}", "I'd like to leave on the {travelDate}", "I'd like to fly out of {fromCity}" ]
        }
      ]
    }
  }
}
```
Alexa Conversations (new in 2020)

When you build an Alexa Conversations skill, you create the following components that train Alexa Conversations how to interact with your user.
Intents / ice_cream

Sample Utterances (15)

What might a user say to invoke this intent?

- i would like {num_scoops} scoops of {flavor} in a {container} with {toppings_one} and {toppings_two}
- I would like {num_scoops} scoops of {flavor} in a {container} with {toppings_one} please
- a {container} with {num_scoops} scoops of {flavor}
- one ice cream please
- a {container} with {flavor} {num_scoops} scoops