DIALOGUE STATE TRACKING AND EVALUATION

Presented By: Kshitij Mishra (Ph.D. Scholar, CSE Department, IIT Patna)
Research Area: Personalization in Conversational AI

Supervised By: Dr. Asif Ekbal

Dialogue State Tracking

Dialogue State Tracking

- A **dialogue-state tracker** determines both the ***current state*** of the frame (the fillers of each slot), as well as the ***user's most recent dialogue act.***
- Thus, a dialogue-state 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 shows the required output of the dialogue state tracker after each turn:

User: I'm looking for a cheaper restaurant
`inform(price=cheap)`

System: Sure. What kind - and where?

User: Thai food, somewhere downtown
`inform(price=cheap, food=Thai, area=centre)`

System: The House serves cheap Thai food

User: Where is it?
`inform(price=cheap, food=Thai, area=centre); request(address)`

System: The House is at 106 Regent Street

Dialogue State Tracking

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

  ‘I’d like Cantonese food near the Mission District’

  has the structure

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

- Simplest dialogue state tracker might just take the output of a slot-filling sequence-model after each sentence.
- Complex model can make use of the reading-comprehension architectures such as training a classifier for each slot to decide whether its value is being changed in the current sentence or should be carried over from the previous sentences.
  - If the slot value is being changed, a span-prediction model is used to predict the start and end of the span with the slot filler.

Dialogue Policy

- The goal of the **dialogue policy** is to ***decide what action the system should take next,*** that is, what ***dialogue act*** to generate.
- Formally, at turn *i* in the conversation we want to ***predict*** which action *A_i* to take, based on the entire dialogue state.
- The state could mean the entire sequence of dialogue acts from the system (A) and from the user (U), in which case the task would be to compute:

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

- Simplifying this by maintaining as the dialogue state mainly just the set of slot-fillers that the user has expressed.
- Such a policy might then ***just condition*** on the ***current dialogue state*** as represented just by the current state of the frame *Frame_i* (which slots are filled and with what) and the last turn by the system and user:

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

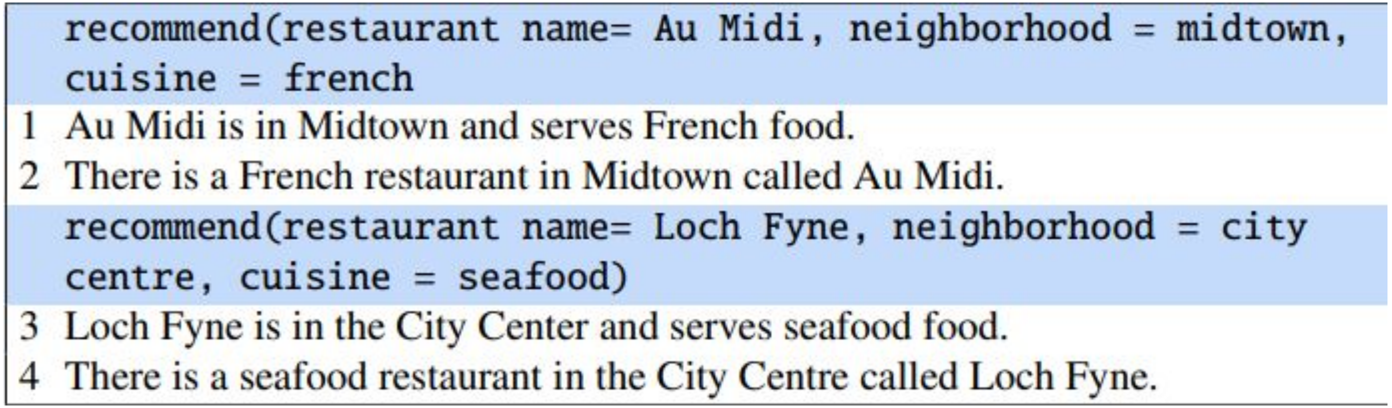
Dialogue Policy

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

Natural language generation in the dialogue-state model

- The task of **natural language generation (NLG)** in the information-state architecture is often modeled in two stages,
    - **content planning** (***what to say***), and
    - **sentence realization** (***how to say it***).
- Content planning has been done by the dialogue policy, which has ***chosen the dialogue act to generate*** and chosen some attributes (**slots and values**) that the planner wants to say to the user.
- The goal of the sentence realizer is to generate a sentence consistent with ongoing dialogue flow by training on many such examples of representation/sentence pairs from a large corpus of labeled dialogues.

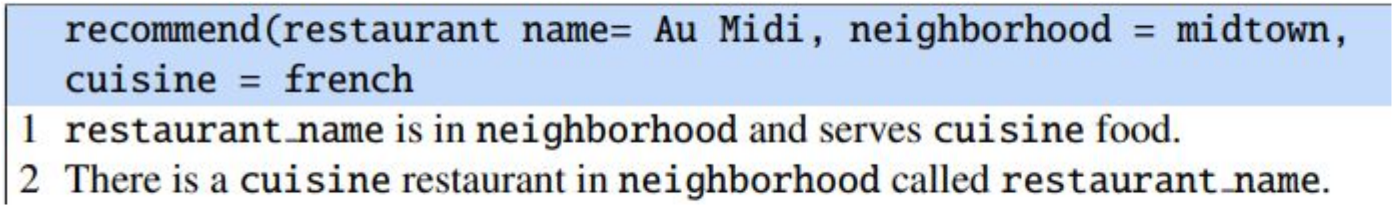
Natural language generation in the dialogue-state model



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

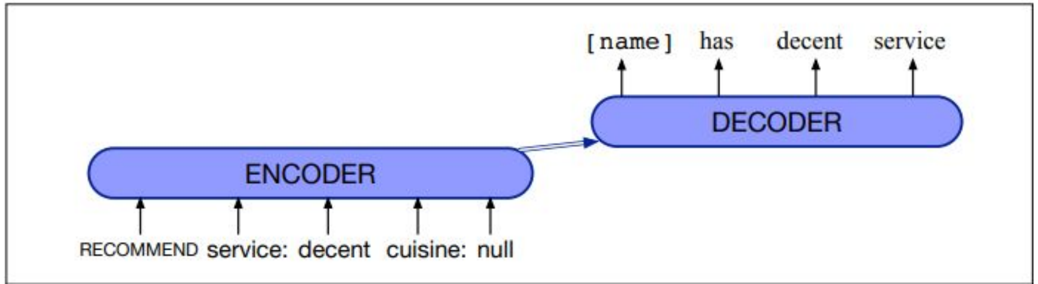
- The content planner has chosen the dialogue act RECOMMEND and some particular slots (name, neighborhood, cuisine) and their fillers.
- Then sentence realizer generates a sentence like lines 1 or 2.
- **Challenge: Training data is hard to come by.** We are unlikely to see every possible restaurant with every possible attribute in many possible differently worded sentences.
- Therefore it is common in sentence realization to increase the generality of the training examples by ***delexicalization.***

DST-NLG: Delexicalization



	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.

- **Delexicalization** is the process of ***replacing specific words*** in the training set that ***represent slot values*** with a ***generic placeholder token*** representing the slot.
- Mapping from frames to delexicalized sentences is generally done by encoder decoder models, trained on large hand-labeled corpora of task-oriented dialogue.
- The input to the encoder is a sequence of tokens *x_t* that represent the dialogue act and its arguments.
- Thus the dialogue act RECOMMEND and the attribute/value pairs **`service:decent, cuisine:null`** might be represented as a flat sequence of tokens each mapped to a learned embedding *w_t*.



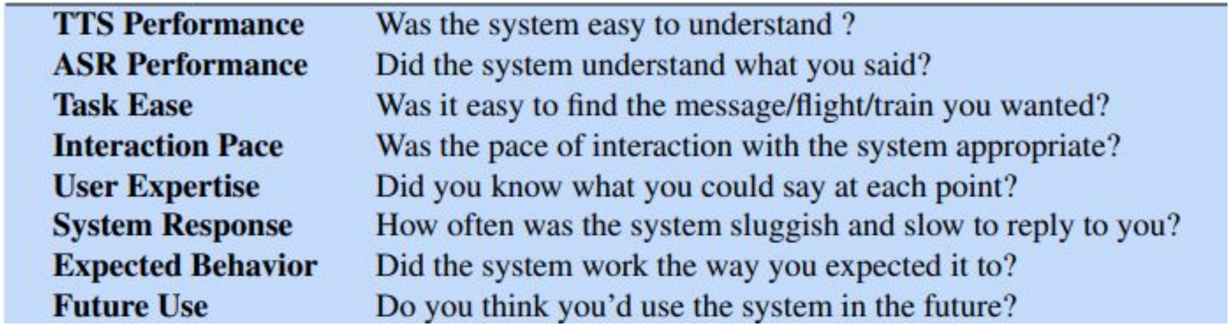
Evaluating Dialogue Systems

Evaluating Dialogue Systems

- Since chatbots and task-oriented systems have different goals, hence they are evaluated differently:
    - task-based systems have to complete a task like booking a flight.
    - chatbots have a different kind of goal, like being enjoyable to users.
- Evaluating Task-oriented Dialogue Systems:
    - if the task is unambiguous, simply measure absolute task success
- Evaluating Chatbots:
    - generally evaluated by humans, who assign a score.
    - can be the human who talked to the chatbot (**participant evaluation**) or a third party who reads a transcript of a human/chatbot conversation (**observer evaluation**).

Evaluating Task-oriented Dialogue Systems

- **User satisfaction rating:** having users interact with a dialogue system to perform a task and then having them complete a questionnaire.



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?

  - For example, sample sample multiple-choice questions (shown in above figure) responses can be mapped into the range of 1 to 5, and then averaged over all questions to get a total user satisfaction rating.

- It is often economically infeasible to run complete user satisfaction studies after every change in a system.
  - **Solution:** *performance evaluation heuristics that correlate well with human satisfaction*

Evaluating Task-oriented Dialogue Systems

- **Performance evaluation heuristics:**
  - **Task completion success:** evaluating the correctness of the total solution (can be **slot error rate** - the percentage of slots that were filled with the correct values).
    - **Slot Error Rate** for a Sentence = $\frac{\text{\# of inserted/deleted/subsituted slots}}{\text{\# of total reference slots for sentence}}$
    - **Task error rate:** quantify how often the goal of the task is completed or not.
  - **Efficiency cost:** measures of the system's efficiency at helping users.
    - can be measured by the total elapsed time for the dialogue in seconds, the number of total turns or of system turns, or the total number of queries
  - **Quality cost:** measures other aspects of the interactions that affect user's perception of the system
    - the number of times the user had to barge in (interrupt the system)
    - the number of time-out prompts played when the user didn't respond quickly enough.

Evaluating Chatbots

- In the **participant evaluation**, the human evaluator can chat with the model for some number of turns (6-10) and then rates (1-5 scale) the chatbot on several dimensions capturing conversational quality such as:
  - **avoiding repetition:** How repetitive was this user?
  - **interestingness:** If you had to say one of these speakers is interesting and one is boring, who would you say is more interesting?
  - **making sense:** How often did this user say something which did NOT make sense?
  - **engagingness:** How much did you enjoy talking to this user?
  - **knowledgeable:** If you had to say that one speaker is more knowledgeable and one is more ignorant, who is more knowledgeable?
- Some other evaluation dimensions can be: ***fluency, listening, inquisitiveness, consistency, adequacy, humanness*** etc.

Automatic chatbot evaluation limitations

- Computational measures of generation performance like BLEU or ROUGE or embedding dot products between a chatbot’s response and a human response correlate very poorly with human judgments.

- Performs poorly because there are so many possible responses to any given turn.

- Simple word-overlap or semantic similarity metrics work best when the space of responses is small and lexically overlapping, which is true of generation tasks like machine translation or possibly summarization, but definitely not dialogue.

References

- Jurafsky, Dan, and James H. Martin. *"Speech and language processing. 3rd ed. draft"* US, Jan, 2022.

- https://web.stanford.edu/~jurafsky/slp3/

- Gao, Jianfeng, Michel Galley, and Lihong Li. “*Neural approaches to conversational AI: Question answering, task-oriented dialogues and social chatbots.*” Now Foundations and Trends, 2019.

Thank You

QA