Natural Language Processing

Lecture 14 : Dialogue Systems

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Dialogue systems

J.A.R.V.I.S. in ©Iron Man

Dialogue systems

Dialogue systems communicate with users in natural language (text or speech).

Two main typologies of dialogue systems

- **chatbots** mimic open-ended conversation characteristic of human-human interaction.

  GPT-4 Omni (OpenAI), Copilot (Microsoft), Gemini (Google), BlenderBot (Meta).

- **digital assistants** engage conversation to help users accomplish domain-specific tasks.

  Siri (Apple), Alexa (Amazon), Home (Google), etc.; also called virtual assistants.

Human Conversation

Oleg Lekhnitsky on Unsplash

Human Conversation

Conversation between humans is an intricate and complex **joint** activity.

A **dialogue** is a sequence of turns. Each **turn** is a single contribution from one speaker.

C(lient)$_1$:	... I need to travel in May.
A(gent)$_2$:	And, what day in May do you want to travel?

A system has to know when to stop and when to start talking. This task is called **endpoint detection**.

In speech applications, endpoint detection can be quite challenging because of noise and because people often pause in the middle of turns.

Speech acts

Utterances in a dialogue are called **speech acts**.

A speech act can be of four main typologies

- **constative**: answering, claiming, confirming, denying, disagreeing, stating
- **directive**: advising, asking, forbidding, inviting, ordering, requesting
- **commissive**: promising, planning, vowing, betting
- **acknowledgment**: apologizing, greeting, thanking, accepting an acknowledgment

Grounding

A dialogue is a collective act where participants exchange information. To this end, participants need to establish a **common ground**.

In this process, very often the hearer sends acknowledgments that she has understood the speaker. This is called **grounding**.

C_1 :	... I need to travel in May.
A_2 :	And, what day **in May** did you want to travel?
⋮	
C_{19} :	I would consider staying there an extra day til Sunday.
A_{20} :	**OK**. **On Sunday** I have ...

Dialogue structure

Natural conversations have an underlying structure composed by patterns of different types.

Very common patterns are the so-called **adjacency pairs**, such as

- question/answer
- proposal/acceptance
- compliment/downplay
- ...

Dialogue structure

Another common pattern is the **sub-dialogue**, as in the following example (# indicates overlapping speech)

A_{16} :	OK. There's #two non-stops ...#
C_{17} :	#Act...actually #, what day of the week is the 15th?
A_{18} :	It's a Friday.
C_{19} :	Uh hmm. I would consider staying there an extra day til Sunday.
A_{20} :	OK...OK. On Sunday I have ...

Sub-dialogues are not recursive, finite-state automata can be used to model dialogue structure.

Inference

Sometimes the speaker does not respond directly to the question, but rather expects the hearer to draw certain **inferences** leading to the answer.

A_2 :	And, what day in May did you want to travel?
C_3 :	OK uh I need to be there for a meeting that's from the 12th to the 15th.

Chatbots

James Yarema on Unsplash

Chatbots

Three main classes of chatbots

- rule-based systems

  mainly of historical interest
- corpus-based systems
- hybrid systems

Rule-based systems

Rule-based systems use hand-written **regular expressions**.

A very influential rule-based system is **Eliza** from the late ‘60, designed to simulate conversation with a psychologist.

ELIZA:	Is something troubling you?
YOU:	Men are all alike.
ELIZA:	What is the connection, do you suppose?
YOU:	They are 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 am depressed much of the time.
ELIZA:	I am sorry you are depressed.
YOU:	It’s true. I am unhappy.
ELIZA:	Can you explain what made you unhappy?

Corpus-based systems

Corpus-based systems mine large datasets of human-human conversations.

These systems are enormously data-intensive, requiring hundreds of millions of word tokens for pretraining.

Commonly used techniques:

- use pseudo-conversation datasets such as Twitter, Reddit, Weibo
- extract responses from knowledge sources such as Wikipedia

Once a chatbot has been put into practice, the acquired human turns can be used as additional data for **fine tuning**.

Corpus-based systems

Corpus-based systems mostly generate a **single response** turn that is appropriate given the entire conversation.

Two main **neural** techniques to provide a response to a user turn

- **response by retrieval**, based on neural information retrieval
- **response by generation**, based on encoder-decoder architectures or large language models

Response by retrieval

Let C be a corpus of conversations. The **main idea** is to

- view a user turn as a query q
- retrieve from C the response r^* that is most pertinent to q

Adopt a **bi-encoder model**:

- use two separate encoders for q and r
- train encoders in such a way that some **similarity function** pairs q to the proper r
- for inference, map q and search for optimal r using the similarity function

Response by retrieval

We can implement this through BERT and [CLS] token.

For **inference**

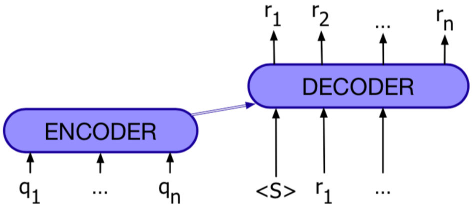
$$
\begin{aligned}
\mathbf{h}_q &= \mathrm{BERT}_Q(q)[\mathrm{CLS}] \\
\mathbf{h}_r &= \mathrm{BERT}_R(r)[\mathrm{CLS}] \\
r^* &= \operatorname*{argmax}_{r \in C} \mathbf{h}_q \cdot \mathbf{h}_r
\end{aligned}
$$

Can be extended using a longer context for q, up to the whole preceding conversation.

Response by generation

The **main idea** is to think of response production as an encoder-decoder task, transducing from the user's prior turn to the system's turn.

Assume query $q = q_1, \dots, q_n$ and response $r = r_1, \dots, r_m$



Response by generation

System can be improved by

- using beam search to avoid repetitive answers such as "I'm OK"

  Recall that each output token r_t is selected using a probability distribution.

- adding minimum length constraints on the answers

Corpus-based systems

Corpus-based chatbots can be much more interesting if they are combined with question answering methods.

Example : We need to be able to respond to user's turns like "Tell me something about Beijing".

This requires one of the following two

- training on text knowledge sources other than dialogue, using techniques from previous lecture
- fine-tune a large language model on a conversational dataset and use the language model directly as a response generator

  Implementation of instructed LLM, through reinforcement learning with human feedback.

Hybrid systems

Chatbots can also be built with architectures that are **hybrids** of the rule-based and corpus-based systems.

A typical architecture consists of several generators, using techniques such as

- fine-tuned neural large language models
- neural paraphrase models tuned on Wikipedia for question answering
- handwritten templates to generate utterances triggered by user's turns

Each response generators can specify a **priority**, and a ranker is used to pass control between response generators.

Research papers

Title: Emora: An Inquisitive Social Chatbot Who Cares For You
Authors: Sarah E. Finch, James D. Finch, Ali Ahmadvand, Ingyu (Jason)Choi, Xiangjue Dong, Ruixiang Qi, Harshita Sahijwani, Sergey Volokhin, Zihan Wang, Zihao Wang, Jinho D. Choi
Repository: arXiv.org.cs.Computation and Language, 10 Sep 2020
Content: This work presents a social chatbot that aims to bring experience-focused interaction to the current field of conversational AI. New conversational abilities are developed that support dialogues that consist of a collaborative understanding and learning process of the partner's life experiences.

https://arxiv.org/abs/2009.04617

Research videos

Title: Emora at DataBlitz 2021
Content: The NLP group at Emory University lead by professor Jinho Choi won the 3rd edition of the Alexa Prize Socialbot Grand Challenge. This video reports a quick overview of the chatbot architecture.

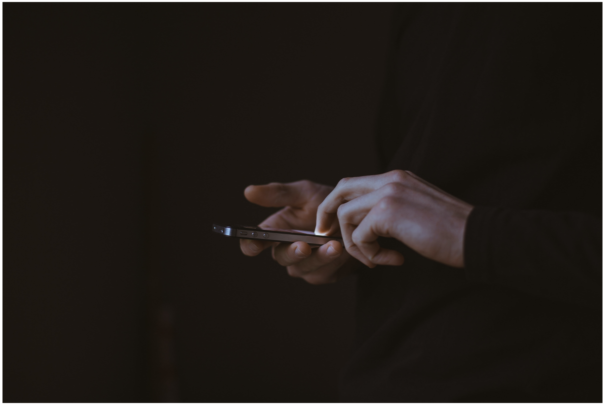
https://www.youtube.com/watch?v=1lwefQ3GnWw

Research papers

Title: Neural Generation Meets Real People: Towards Emotionally Engaging Mixed-Initiative Conversations
Authors: Ashwin Paranjape, Abigail See, Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, Christopher D. Manning
Repository: arXiv.org.cs.Computation and Language, 5 Sep 2020
Content: This work presents an open-domain dialogue agent that provides a responsive, personalized user experience, capable of talking knowledgeably about a wide variety of topics, as well as chatting empathetically about ordinary life.

https://arxiv.org/abs/2008.12348

Digital assistants



Gilles Lambert from Unsplash

Digital assistants

Digital assistants have the goal of helping a user to solve specific tasks, such as making airplane reservation or buying a product.

Also called task-based dialogue systems.

Today, almost all companies have digital assistants to engage their users and serve customers by catering to their queries.

As per a report by Gartner, in year 2020 this was estimated to 85% of the customer service and 80% of businesses.

Also used as reception assistant, help desk assistant, tutor or teacher.

Digital assistants

We distinguish two main architectures for digital assistants

- **frame-based architecture**: one of the very early architectures, still in use in medium-scale systems
- **dialogue-state architecture**, more advanced, used in modern, large-scale industrial systems

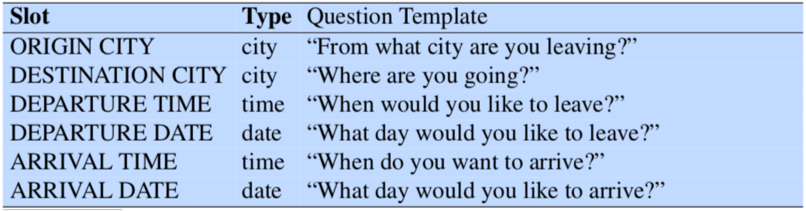
Frame-based dialogue systems

picture here

Frame-based dialogue systems

A **frame** is a kind of knowledge structure representing information and intentions that the system needs to extract from user's sentences, and is defined as a collection of **slot**, **value** pairs.

Example : In the travel domain, a **flight frame** may have the form



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 might be frames themselves, composing a type hierarchy.

In the example above, type date requires a frame.

Frame-based dialogue systems

The system goal is to **fill the slots** in the frames with the appropriate values.

To do this, the system uses

- pre-specified **question templates** associated with each slot in each frame
- associated **production rules** to extract information from answers

Frame-based dialogue systems

Production rules may also fill several **related slots** from different frames.

Example : A rule associated with the DESTINATION slot in the plane booking frame might automatically enter the city as the default STAYLOCATION in the hotel booking frame.

A user turn might also provide information for **multiple slots**. The system must be able to detect these cases and correctly fill the relevant slots.

Example : I want a flight from San Francisco to Denver one way, leaving after 5pm on Tuesday.

Frame-based dialogue systems

The system must be able to switch dialogue control **dynamically**, on the basis of the acquired information, skipping questions associated with already filled slots.

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

Frame-based dialogue systems

There are three general tasks that need to be solved in frame-based dialogue systems

- **Domain classification**: in case of multi-domain dialogue systems, detect the appropriate domain.

  The user might want to deal with email, calendar, clock, etc.

- **Intent determination**: given the domain, which goal is the user trying to accomplish?

  For home banking, we might have money transfer, balance checking, etc.

- **Slot filling**: extract the particular slots and fillers needed to carry out the user intent.

Frame-based dialogue systems

Example :
Show me morning flights from Boston to San Francisco on Tuesday.

The system might build the following representation

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

Frame-based dialogue systems

Hand-written rules were proposed in the 70's and are still in use for intent determination and slot filling in industrial applications.

Rules are defined by means of **regular expressions**.

Alternatively, hand-designed **semantic grammars** can be used, consisting in context-free grammar rules in which the left-hand side corresponds to the semantic entity being expressed.

Examples in next slide.

Semantic grammars

Example : The context-free grammar rules

$$\text{SHOW} \rightarrow \text{show me} \mid \text{I want} \mid \text{can I see} \mid \cdots$$

are used to detect an intent of type SHOW.

Example : The context-free grammar rule

$$\text{ORIGIN} \rightarrow \text{from CITY}$$

is used to detect the slot ORIGIN-CITY for the AIR-TRAVEL frame. Here CITY is a nonterminal for some other rules.

Semantic grammars can be parsed by any CFG parsing algorithm.

Dialogue-state architecture

picture here

Dialogue-state architecture

Dialogue-state architecture is a more advanced version of the frame-based architecture.

Dialogue-state systems can be text-based or speech-based.

Modern commercial dialogue systems are **hybrids**, using frame-based architecture augmented with some of the components of the dialogue-state architecture.

Dialogue-state architecture

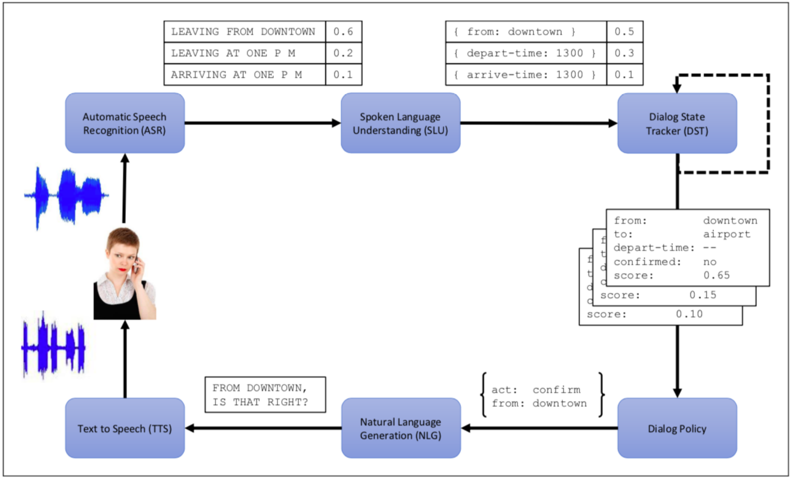
A typical dialogue-state system is based on **six components**

- automatic speech recognition
- natural language understanding
- dialog state tracker
- dialog policy
- natural language generation
- text to speech

Here we **do not deal** with automatic speech recognition component and text to speech component.

Dialogue-state architecture

Six components of a typical dialogue-state system



LEAVING FROM DOWNTOWN 0.6
LEAVING AT ONE P M 0.2
ARRIVING AT ONE P M 0.1
{ from: downtown } 0.5
{ depart-time: 1300 } 0.3
{ arrive-time: 1300 } 0.1
Automatic Speech Recognition (ASR)
Spoken Language Understanding (SLU)
Dialog State Tracker (DST)
from: downtown
to: airport
depart-time: --
confirmed: no
score: 0.65
score: 0.15
score: 0.10
act: confirm
from: downtown
FROM DOWNTOWN, IS THAT RIGHT?
Text to Speech (TTS)
Natural Language Generation (NLG)
Dialog Policy

Dialogue-state components

The **natural language understanding** component extracts slot fillers from the user utterances using machine learning.

Frame-based architectures use hand-written rules.

The **dialogue state tracker** maintains the current state of the dialogue, which includes

- the user most recent dialogue act
- the entire set of slot-filler constraints the user has expressed so far

Dialogue-state components

The **dialogue policy** component decides what to do next: answer a question, ask a clarification, make a suggestion, and so on.

This results in more sophisticated dialogue policies than in frame-based architectures.

The **natural language generation** component can condition on the exact dialogue context, to produce turns that seem much more natural.

In frame-based architectures, all sentences are produced from pre-written templates.

Dialogue act

Dialogue-state systems make use of **dialogue acts**, which

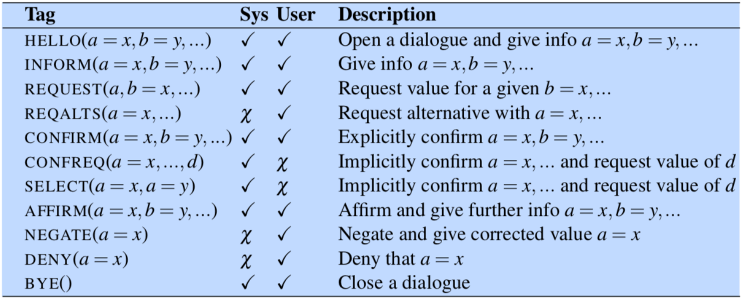
- implement the conversation turn
- carry out the function of speech act and of grounding

Depending on the domain, each system uses a specific set of dialogue act **categories** to classify its dialogue acts.

Dialogue act categories are specific for system turn, user turn, or both.

Dialogue act categories

Example : Dialog act categories for restaurant recommendation system



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

Dialogue act categories

Example : Dialog act categories for restaurant recommendation system

S: You are looking for a restaurant. What type of food do you like?
`confreq(type = restaurant, food = ??)`

U: I'd like an Italian somewhere near the museum.
`inform(food = Italian, near = museum)`

Natural language understanding

This component exploits **sequence labeling** and **sentence classification** techniques from previous lectures to solve the following three tasks

- domain classification
- intent extraction
- slot filling

We need a **training set** that associates each sentence with the correct domain, intent, and set of slots.

Training sets are often bootstrapped using rule-based systems, and then carefully hand-checked.

Natural language understanding

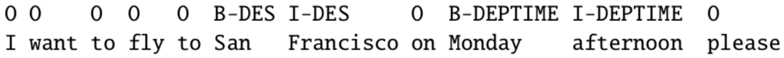
Simple neural architecture

- exploit contextual embedding models (for instance BERT) to compute input sentence representation
- use classifier at the `<EOS>` token to determine **domain** and **intent**
- use a feedforward layer and a simple 1-of-N classifier to determine token labels for **slot filling**

  This is the simplest NN architecture proposed for sequence labeling. Alternatively, you may consider LSTM coupled with CRF.

Natural language understanding

Example : Slot filling can be treated using BIO labels and appropriate entities



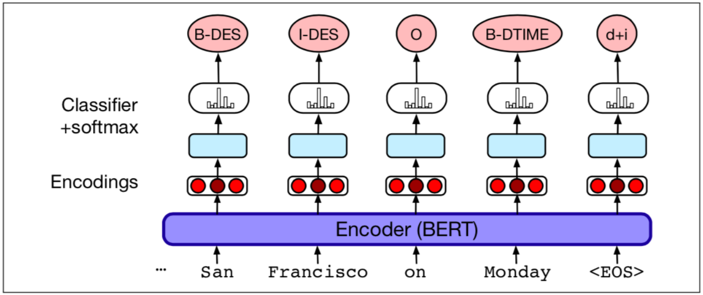
```
O O    O  O   O  B-DES I-DES     O  B-DEPTIME I-DEPTIME  O
I want to fly to San   Francisco on Monday    afternoon  please
```

Slot-fillers must later be normalized to the correct form in the ontology.

In the example above, San Francisco must be replaced by SFO.

Natural language understanding

Putting everything together we have



Dialog state tracker

A **dialogue state** consists of

- the entire frame at a given point of the dialogue
- the user's most recent dialogue act

The **dialogue state tracker** computes the current dialogue state, that is

- updates the running frame
- classifies the user's most recent dialogue act

Dialogue act detection and slot filling are complementary to each other and are often performed **jointly**.

Dialog state tracker

Example : Output of the dialogue state tracker after each user turn.

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

inform used here to represent the running frame.

Dialog state tracker

Dialogue act classification is based on embeddings representing the current input sentence and the prior dialogue acts.

Frame update is based on reading-comprehension techniques from previous lecture.

Use **supervised** classification / information extraction, trained on hand-labeled dialog acts.

Dialog state tracker

If a dialogue system misunderstands an utterance, the user will generally correct the error by reformulating the utterance. This is called a **correction act**.

Dialog state tracker is also in charge of detecting correction acts, and interacts with slot filling to decide which slot value is being changed.

To detect correction acts, systems can make use of special features, orthogonal to contextual embedding features.

Dialog policy

This component decides what dialogue act the system should generate at step i, based on the entire dialogue state

Let A_j be the act from the system and U_j be the act from the user, both at step j. We make the following **assumption**

$$\begin{aligned}\hat{A}_i &= \operatorname*{argmax}_{A_i \in \mathcal{A}} P(A_i \mid A_1, U_1, \ldots, A_{i-1}, U_{i-1}) \\ &= \operatorname*{argmax}_{A_i \in \mathcal{A}} P(A_i \mid \mathrm{Frame}_{i-1}, A_{i-1}, U_{i-1})\end{aligned}$$

The chosen dialog act might also contain slot values, which come from training text or else are retrieved from some database.

Name and address of restaurant, bank account, etc.

Dialog policy

Probabilities can be estimated by a neural classifier, using neural representations of the slot fillers and the utterances.

Training is performed using **reinforcement learning**: the system gets a positive reward if the dialogue policy terminates with the correct slot representation, and a negative reward otherwise.

Dialog policy

This component also implements specialized strategies to deal with system mistakes in interpretation of the user's input.

Explicit confirmation: the system asks the user a direct question to confirm the system's understanding.

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

Dialog policy

Implicit confirmation: the system can demonstrate its understanding as a grounding strategy.

U:	I want to travel to Berlin
S:	When do you want to travel **to Berlin**?

Explicit confirmation is safer, implicit confirmation is more fluent in conversation.

Dialog policy

Rejection: the system gives the user a prompt like "I'm sorry, I didn't understand that"

When an utterance is rejected, systems often follow a strategy of **progressive prompting**

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

Natural language generation

This component generates the text of a response to the user, once the policy has decided what speech act to generate.

This task is modelled in two stages

- **content planning**: what to say
- **sentence realization**: how to say it

Content planning is best implemented within the dialogue policy, which chooses the dialogue act and the slot values.

Natural language generation

Sentence realization translates from the dialogue act and its arguments to text sentences.

Example :

```
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.

Natural language generation

Sentence realizer is trained on representation/sentence pairs from a large corpus of labeled dialogues.

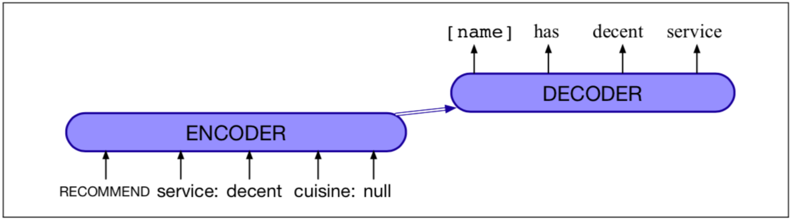
To increase the generality of the training examples one must use **delexicalization**, replacing specific slot values with a generic placeholder.

Example :
`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`

Natural language generation

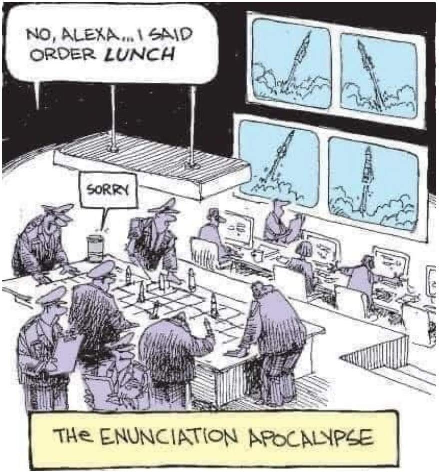
Translation from frames to delexicalized sentences is generally implemented by encoder decoder models.

The input is the dialog act with **attribute/value** pairs. Each token is then embedded and translated.


[name] has decent service
DECODER
ENCODER
RECOMMEND service: decent cuisine: null

Finally, we use the input frame to **relexicalize** (fill in the exact slot value).

Evaluation


NO, ALEXA... I SAID ORDER LUNCH
SORRY
THE ENUNCIATION APOCALYPSE

Evaluation

Evaluation is crucial in dialogue system design.

Chatbots and task-based digital assistants have different goals:

- chatbots have to be enjoyable to users
- digital assistants have to complete a task

These systems are therefore evaluated differently.

Chatbot evaluation

Chatbots are evaluated by humans, who assign a score.

Participant evaluation: evaluation is carried out by the human who talked to the chatbot

Observer evaluation: evaluation is carried out by a third party who reads a transcript of a human/chatbot conversation

Chatbot evaluation

Several dimensions are used, capturing conversational quality:

- how fluent is the conversation with the system?
- how repetitive is the system?
- how coherent are conversation turns?
- did the system say something which did not make sense?
- engagingness: how much did you enjoy talking to the system?

Chatbot evaluation

Automatic evaluation methods for chatbots perform poorly because there are so many possible responses to any given turn.

Simple word-overlap measures or semantic similarity metrics, such as those used in machine translation or automatic summarization, do not work well with dialogue.

Digital assistant evaluation

Digital assistant can be **extrinsically** evaluated by measuring the success of task completion.

For a frame-based architecture, one can use **slot success rate**, which is the percentage of slots filled with the correct values.

A less fine-grained measure is **task success rate**, which is the percentage of target tasks correctly achieved.

Did the system book the right plane flight? Did the system put the right event on the calendar?

Ethical issues

picture here

Ethical issues

Machine learning systems replicate **biases** that occur in the training data. This is especially relevant for chatbots.

Microsoft's 2016 Tay chatbot was taken offline 16 hours after it went live, when it began posting messages with racial slurs, conspiracy theories, and personal attacks on its users.

Tay had learned these biases and actions from users who were **adversarially** attacking the system, purposely teaching it to repeat this kind of behaviour.

Systems that interact with users must be designed to be robust to adversarial attacks.

Ethical issues

Corpora drawn from social media like Twitter and Reddit contain examples of hate speech, offensive language, and bias.

Researchers have found that transformer-based generator models **amplify** gender bias in training dialogues.

Addressing these problem by investigating debiasing methods is an important current research goal.

Ethical issues

Privacy is another important ethical issue.

Home dialogue agents may accidentally record a user revealing private information, and the data may later be used to train a conversational model.

Chatbots trained on transcripts of human-human or human-machine conversation must **anonymize** personally identifiable information.