COMP SCI 7327 Concepts in Artificial Intelligence & Machine Learning -Dialogue systems and Chatbots

By Dr Wei Zhang

Outline

- Conversational Systems
  - Chatbots
  - Task-based Dialogue Agents
    - SIRI, interfaces to cars, robots,
    - booking flights or restaurants

Language Empowering Intelligent Assistant

Apple Siri (2011)

Google Now (2012)
Google Assistant (2016)

Microsoft Cortana (2014)

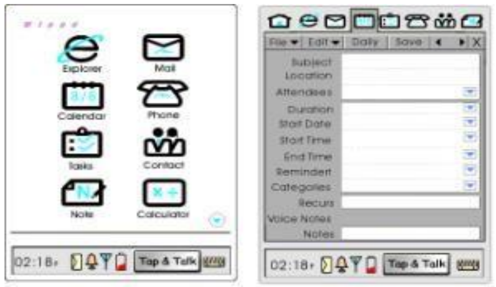
Facebook M & Bot (2015)

Google Home (2016)

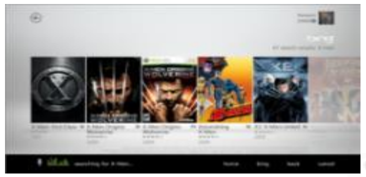
Apple HomePod (2017)

Brief History of Conversatioal Systems

Multi-modal systems
e.g., Microsoft MiPad, Pocket PC

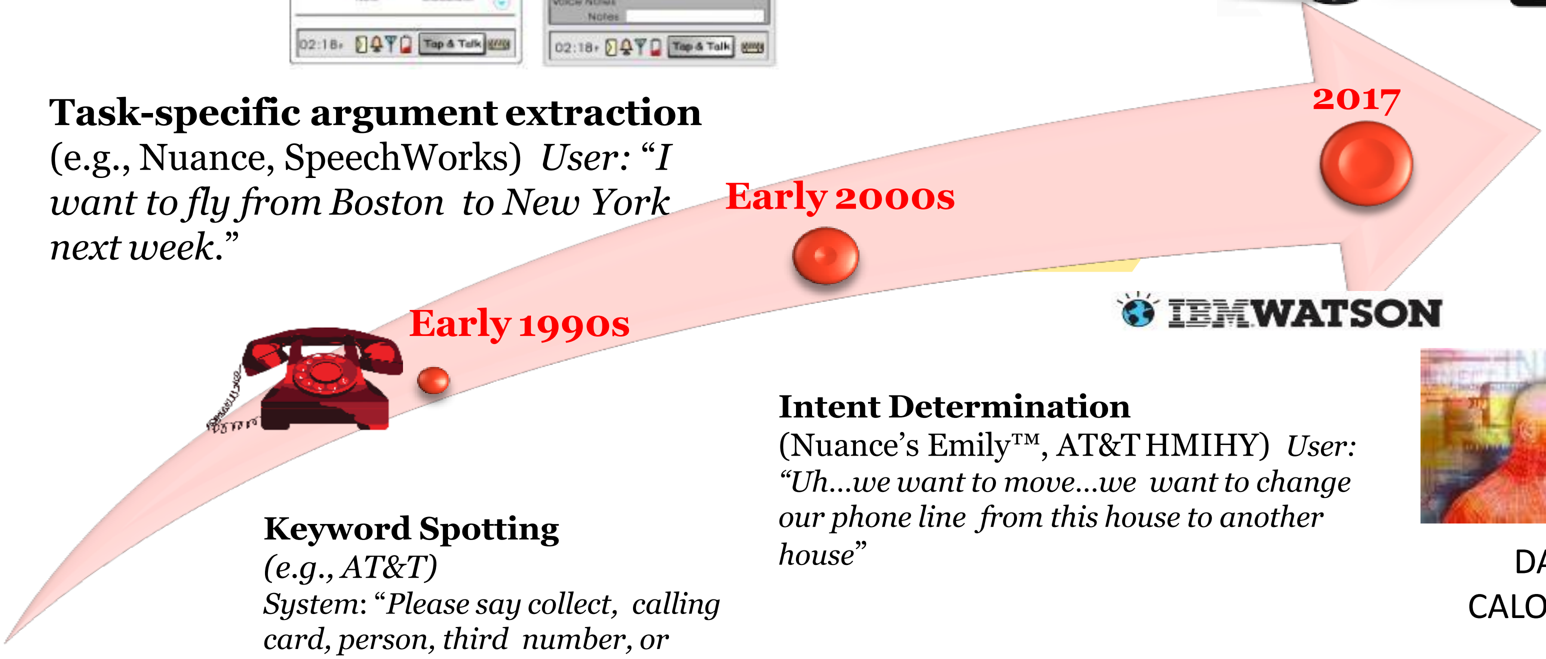


TV Voice Search
e.g., Bing on Xbox



Virtual Personal Assistants

Task-specific argument extraction
(e.g., Nuance, SpeechWorks) *User: "I want to fly from Boston to New York next week."*



Keyword Spotting
(e.g., AT&T)
System: "Please say collect, calling card, person, third number, or operator"

IBM WATSON

Intent Determination
(Nuance's Emily™, AT&T HMIHY) *User: "Uh...we want to move...we want to change our phone line from this house to another house"*

DARPA
CALO Project

Material: http://deepdialogue.miulab.tw

Part I: Chatbots

- Sometimes for fun. But originally designed as a testbed for clinical therapy (And still used that way!)
- Some Chatbots:
  - ELIZA (1966)
  - PARRY (1968)
  - ALICE
  - CLEVER
  - Microsoft Little Bing 小冰

Chatbot Architectures

- Rule-based
  - Pattern-action rules (Eliza)

- Corpus-based (from large chat corpus)
  - Information Retrieval
  - Neural network encoder-decoder

Eliza: Weizenbaum (1966)

Human: Men are all alike.
Eliza: In what way?
Human: They're always bugging us about something or other.
Eliza: can you think of a specific example?
Human: Well, my boyfriend made me come here.
Eliza: your boyfriend made you come here
Human: He says I'm depressed much of the time.
Eliza: I am sorry to here you are depressed.
...
Eliza: Who else in your family takes care of you?
Human: My father
Eliza: Your father
...

Simulate the conversation style as a Rogerian psychologist

How does it work?

- Eliza simulated conversation by using pattern matching and substitution to give users an illusion of understanding on the part of the program.

Eliza pattern/transform rules

`(* YOU * ME)` [*pattern*]

→

`(WHAT MAKES YOU THINK I # YOU)` [*transform*]

* means Kleene *

The # is the constituent in pattern

e.g.,

```
You hate me
WHAT MAKES YOU THINK I HATE YOU
```

NLTK: Eliza

```
# Natural Language Toolkit: Eliza
#
# Copyright (C) 2001-2020 NLTK Project
# Authors: Steven Bird <stevenbird1@gmail.com>
#          Edward Loper <edloper@gmail.com>
# URL: <http://nltk.org/>
# For license information, see LICENSE.TXT

# Based on an Eliza implementation by Joe Strout <joe@strout.net>,
# Jeff Epler <jepler@inetnebr.com> and Jez Higgins <mailto:jez@jezuk.co.uk>.

# a translation table used to convert things you say into things the
# computer says back, e.g. "I am" --> "you are"

from nltk.chat.util import Chat, reflections
```

https://www.nltk.org/_modules/nltk/chat/eliza.html

NLTK: Eliza

```
# a table of response pairs, where each pair consists of a
# regular expression, and a list of possible responses,
# with group-macros labelled as %1, %2.

pairs = (
    (
        r"I need (.*)",
        (
            "Why do you need %1?",
            "Would it really help you to get %1?",
            "Are you sure you need %1?",
        ),
    ),
```

NLTK: Eliza

```
    r"I feel (.*)",
    (
        "Good, tell me more about these feelings.",
        "Do you often feel %1?",
        "When do you usually feel %1?",
        "When you feel %1, what do you do?",
    ),
),
```

NLTK: Eliza

```
(
    r"(.*)\?",
    (
        "Why do you ask that?",
        "Please consider whether you can answer your own question.",
        "Perhaps the answer lies within yourself?",
        "Why don't you tell me?",
    ),
),
(
    r"quit",
    (
        "Thank you for talking with me.",
        "Good-bye.",
        "Thank you, that will be $150.  Have a good day!",
    ),
),
```

NLTK: Eliza

```
(
    r"(.*)",
    (
        "Please tell me more.",
        "Let's change focus a bit... Tell me about your family.",
        "Can you elaborate on that?",
        "Why do you say that %1?",
        "I see.",
        "Very interesting.",
        "%1.",
        "I see.  And what does that tell you?",
        "How does that make you feel?",
        "How do you feel when you say that?",
    ),
),
pairs = (
    (
        r"I need (.*)",
        (
            "Why do you need %1?",
            "Would it really help you to get %1?",
            "Are you sure you need %1?",
        ),
    ),
    (
        r"Why don\'t you (.*)",
        (
            "Do you really think I don't %1?",
            "Perhaps eventually I will %1.",
            "Do you really want me to %1?",
        ),
    ),
    (
        r"Why can\'t I (.*)",
        (
            "Do you think you should be able to %1?",
            "If you could %1, what would you do?",
            "I don't know -- why can't you %1?",
            "Have you really tried?",
        ),
    ),
    (
        r"I can\'t (.*)",
        (
            "How do you know you can't %1?",
            "Perhaps you could %1 if you tried.",
            "What would it take for you to %1?",
        ),
    ),
    (
        r"I am (.*)",
        (
            "Did you come to me because you are %1?",
            "How long have you been %1?",
            "How do you feel about being %1?",
        ),
    ),
    (
        r"I\'m (.*)",
        (
            "How does being %1 make you feel?",
```

```
reflections = {
    "i am": "you are",
    "i was": "you were",
    "i": "you",
    "i'm": "you are",
    "i'd": "you would",
    "i've": "you have",
    "i'll": "you will",
    "my": "your",
    "you are": "I am",
    "you were": "I was",
    "you've": "I have",
    "you'll": "I will",
    "your": "my",
    "yours": "mine",
    "you": "me",
    "me": "you",
}
```

NLTK: Eliza

```python
eliza_chatbot = Chat(pairs, reflections)
```

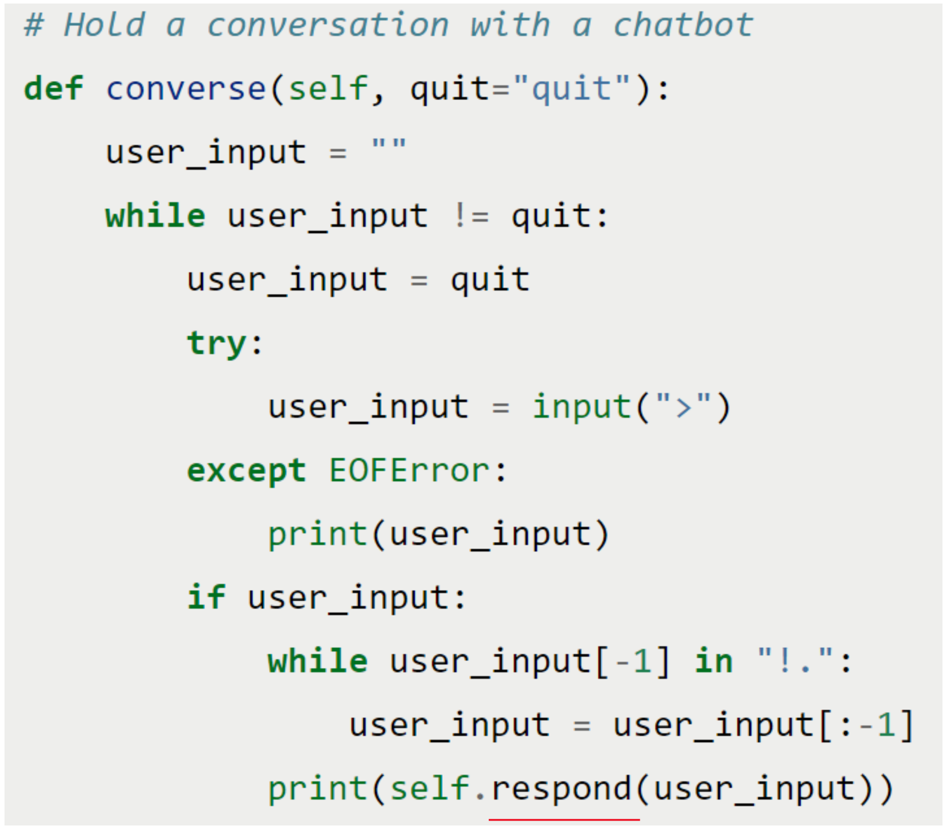
```python
def eliza_chat():
    print("Therapist\n---------")
    print("Talk to the program by typing in plain English, using normal upper-")
    print('and lower-case letters and punctuation.  Enter "quit" when done.')
    print("=" * 72)
    print("Hello.  How are you feeling today?")

    eliza_chatbot.converse()
```

```python
eliza_chat()
```

Module: nltk.chat.util

# https://www.nltk.org/_modules/nltk/chat/util.html



```
# Hold a conversation with a chatbot
def converse(self, quit="quit"):
    user_input = ""
    while user_input != quit:
        user_input = quit
        try:
            user_input = input(">")
        except EOFError:
            print(user_input)
        if user_input:
            while user_input[-1] in "!.":
                user_input = user_input[:-1]
            print(self.respond(user_input))
```

Module: nltk.chat.util

```
def respond(self, str):
    # check each pattern
    for (pattern, response) in self._pairs:
        match = pattern.match(str)

        # did the pattern match?
        if match:
            resp = random.choice(response)  # pick a random response
            resp = self._wildcards(resp, match)  # process wildcards

            # fix munged punctuation at the end
            if resp[-2:] == "?.":
                resp = resp[:-2] + "."
            if resp[-2:] == "??":
                resp = resp[:-2] + "?"
            return resp
```

Module: nltk.chat.util

```python
def _wildcards(self, response, match):
    pos = response.find("%")
    while pos >= 0:
        num = int(response[pos + 1 : pos + 2])
        response = (
            response[:pos]
            + self._substitute(match.group(num))
            + response[pos + 2 :]
        )
        pos = response.find("%")
    return response
```

IR-based Chatbots

Idea: Mine conversations of human chats or human-machine chats, e.g., microblogs, movie dialogs

- Cleverbot
- Microsoft XiaoIce
- Microsoft Tay

Two IR-based Chatbot Architectures

- Return the response to the most similar turn
  - Take user's turn (*q*) and find a (tf-idf) similar turn *t* in the corpus C

    q = "do you like Doctor Who"

    t= "do you like Doctor Strangelove"
  - Grab whatever the response was to *t* as the response to q.

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

- Return the most similar turn

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

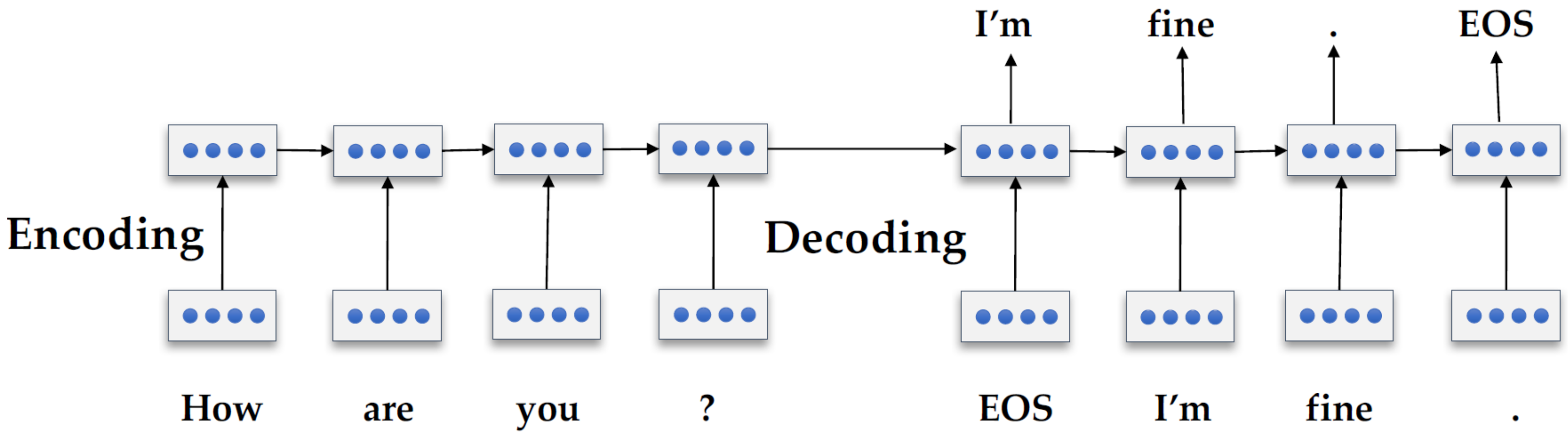
IR-based Models of Chatbots

- Also fine to use other features like user features, or prior turns
- Or non-dialogue text
  - COBOT chatbot (Isbell et al., 2000)
    - sentences from the Unabomber Manifesto by Theodore Kaczynski, articles on alien abduction, the scripts of “The Big Lebowski” and “Planet of the Apes”.
  - Wikipedia text

Neural Chatbots

- Think of response generation as a task of *transducing* from the user's prior turn to the system's turn (map from user1 turn to user2 response).
- Train a deep neural network on
  – movie dialogue databases
  – Twitter conversations
  – ...

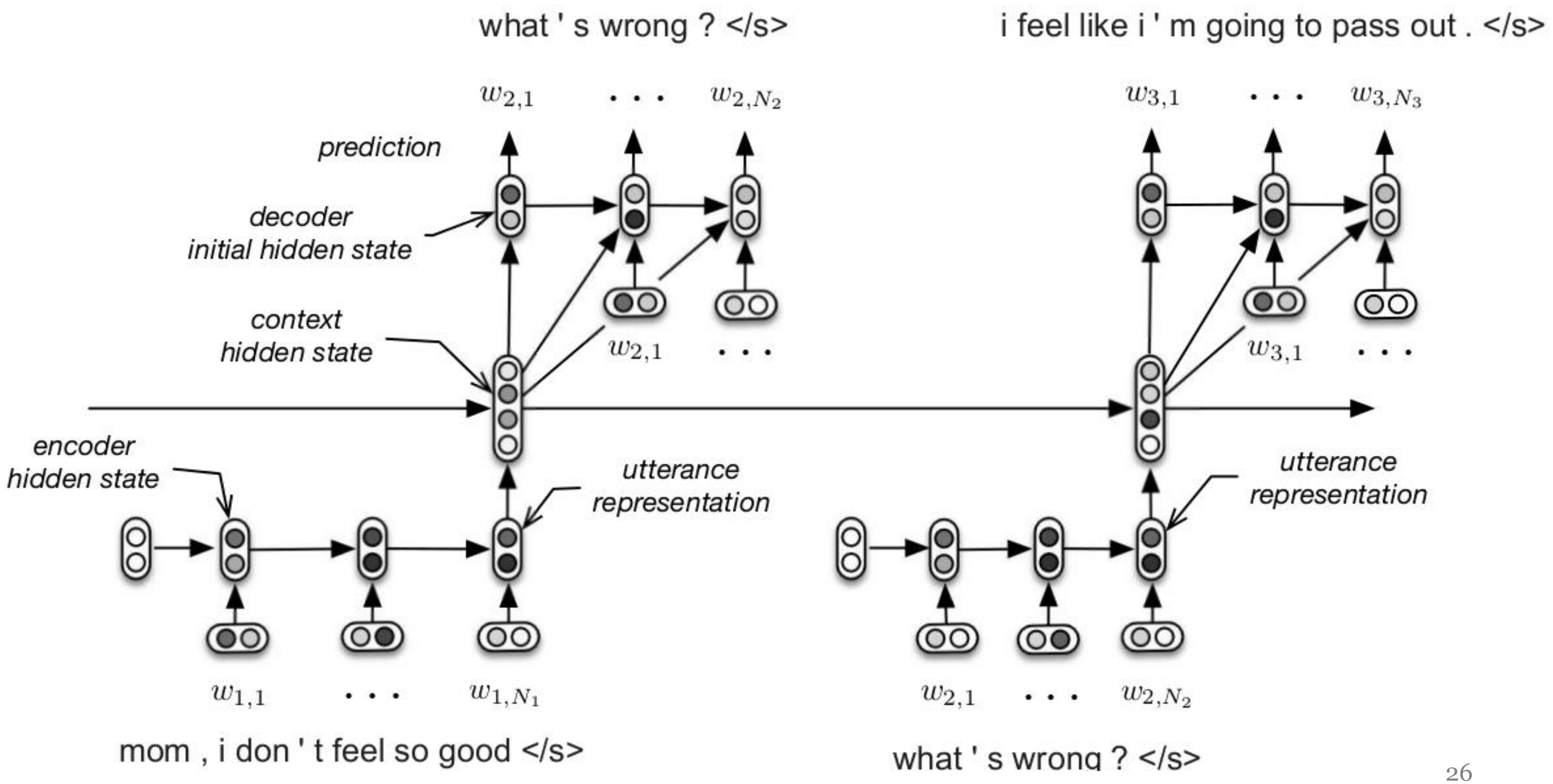
Neural Chatbots



An encoder-decoder model for neural response generation in dialogue.

Neural Chatbots

Serban, Iulian V., Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. "*Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.*"



Part II: Task-based Dialogue Agents

- Sometimes called “frame-based” or “goal-based dialogue agents
- Based on a "domain ontology"
  - A knowledge structure representing user intentions

- One or more **frames**
  - Each a collection of **slots**
  - Each slot having a **value**

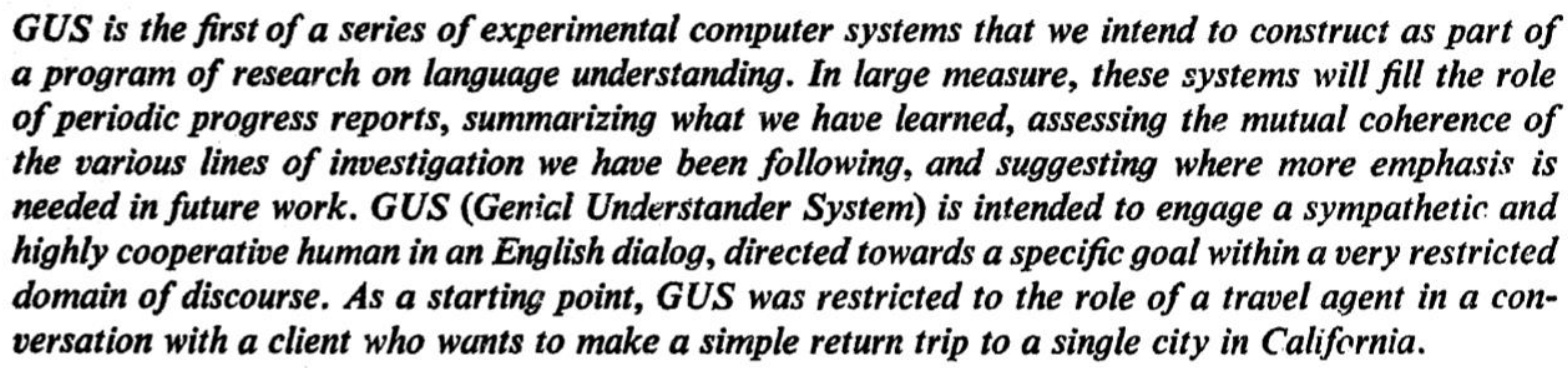
The Frame

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

Slot	**Type**	**Question**
ORIGIN	city	What city are you leaving from?
DEST	city	Where are you going?
DEP DATE	date	What day would you like to leave?
DEP TIME	time	What time would you like to leave?
AIRLINE	line	What is your preferred airline?

Frame-based dialogue agents

- 1977, GUS:



GUS is the first of a series of experimental computer systems that we intend to construct as part of a program of research on language understanding. In large measure, these systems will fill the role of periodic progress reports, summarizing what we have learned, assessing the mutual coherence of the various lines of investigation we have been following, and suggesting where more emphasis is needed in future work. GUS (Genial Understander System) is intended to engage a sympathetic and highly cooperative human in an English dialog, directed towards a specific goal within a very restricted domain of discourse. As a starting point, GUS was restricted to the role of a travel agent in a conversation with a client who wants to make a simple return trip to a single city in California.

Artificial Intelligence Journal, 1977

- Still the industrial state of the art
  – SIRI based on GUS architecture

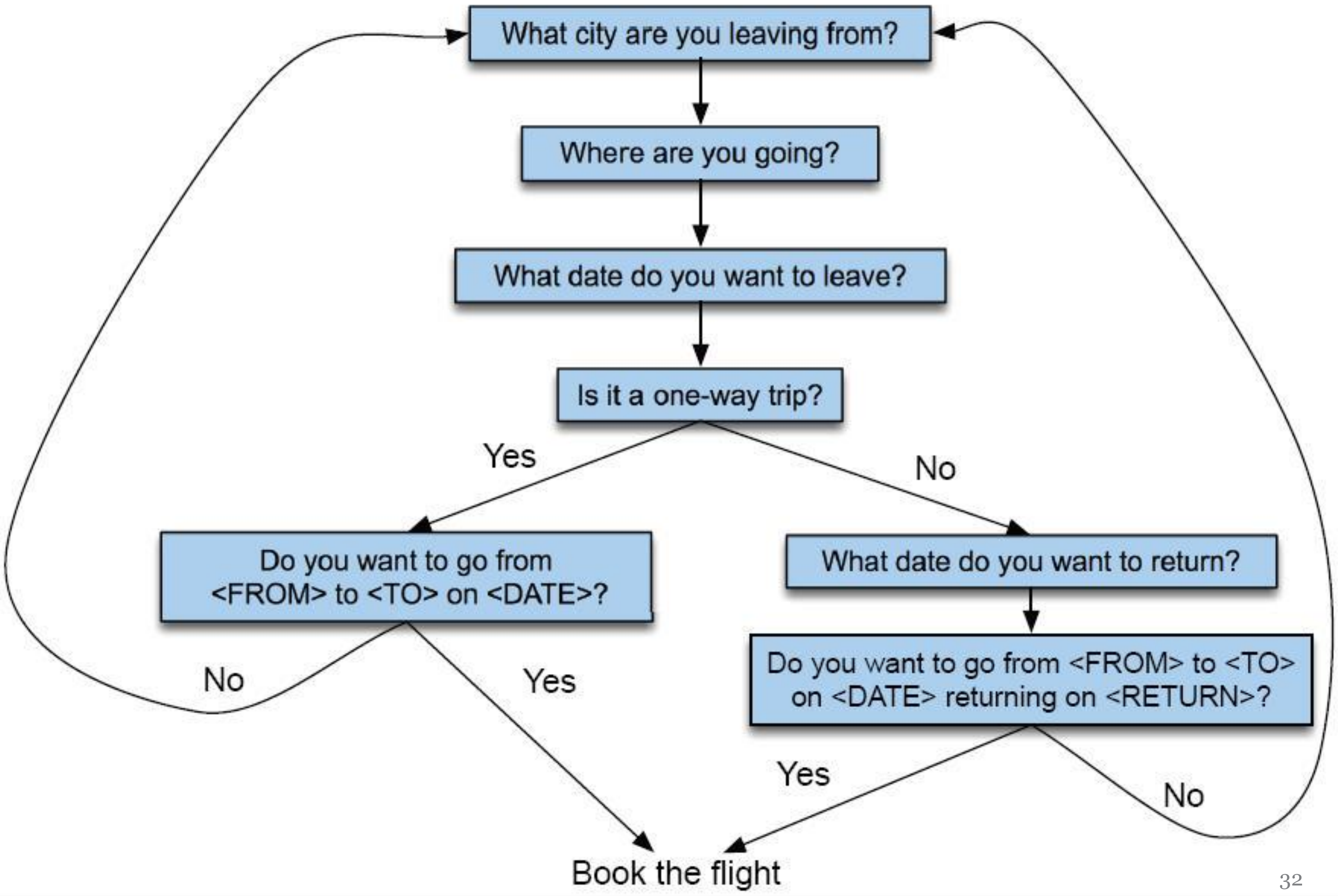
Control Structure for Frame-based Dialogue

- 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

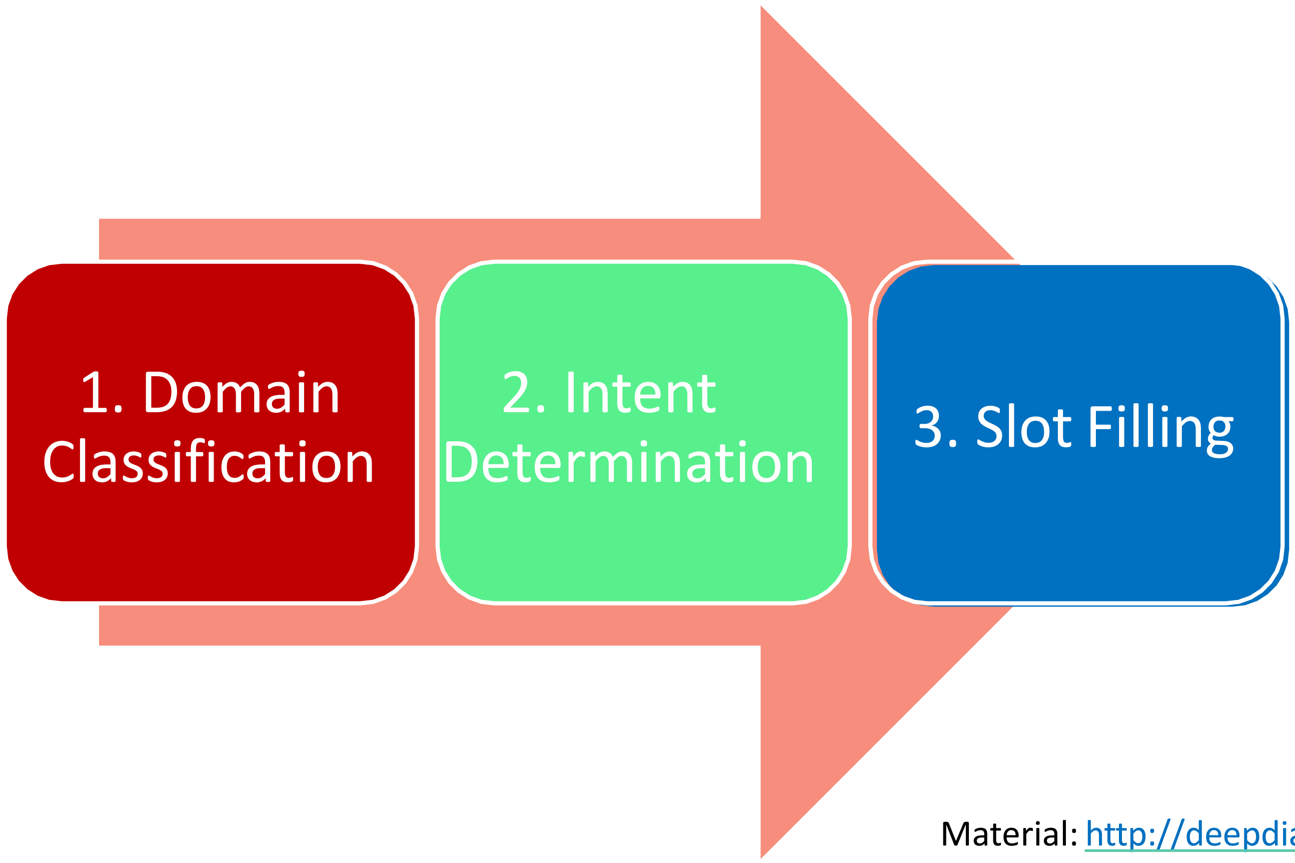
- System completely controls the conversation with the user.
- It asks the user a series of questions
- Ignoring anything the user says that is not a direct answer to the system's questions

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
No
Do you want to go from <FROM> to <TO> on <DATE>?
What date do you want to return?
Do you want to go from <FROM> to <TO> on <DATE> returning on <RETURN>?
No
Yes
Yes
No
Book the flight

NLU for Filling Dialogue Slots

- Pipelined



Material: http://deepdialogue.miulab.tw

NLU for Filling Dialogue Slots –Examples

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
```

NLU for Filling Dialogue Slots – Examples

Wake me tomorrow at six.

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

NLU for Slot-filling

- Rule-based.
- Machine learning (deep learning) based.

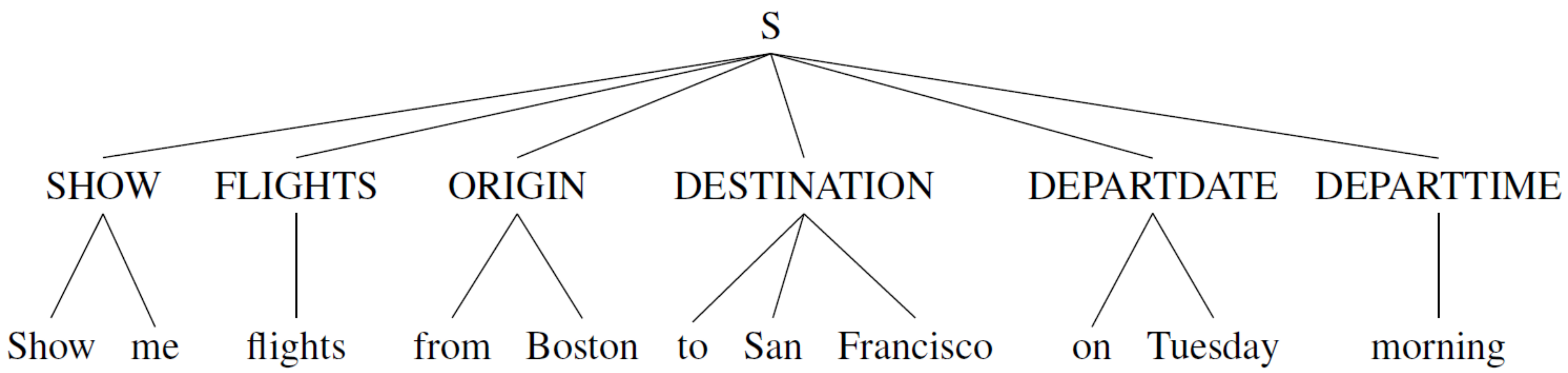
Rule-based Slot-filling

- Rule-based research systems consist of large hand-designed semantic grammars with thousands of rules. A semantic grammar is a context-free 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:

SHOW	$\rightarrow$	show me | i want | can i see|...
DEPART_TIME_RANGE	$\rightarrow$	(after|around|before) HOUR | morning | afternoon | evening
HOUR	$\rightarrow$	one|two|three|four...|twelve (AMPM)
FLIGHTS	$\rightarrow$	(a) flight | flights
AMPM	$\rightarrow$	am | pm
ORIGIN	$\rightarrow$	from CITY
DESTINATION	$\rightarrow$	to CITY
CITY	$\rightarrow$	Boston | San Francisco | Denver | Washington

Rule-based Slot-filling

- Semantic grammars can be parsed by any CFG parsing algorithm resulting in a hierarchical labeling of the input string with semantic node labels, as shown below:



Machine learning based Slot-filling

- Machine learning classifiers to map words to semantic frame-fillers
- Given a set of labeled sentences

```
"I want to fly to San Francisco on Tuesday"
  Destination: San Francisco
  Depart-date: Tuesday
```

  - Build a classifier to map from one to the answer
- Requirements: Lots of labeled data

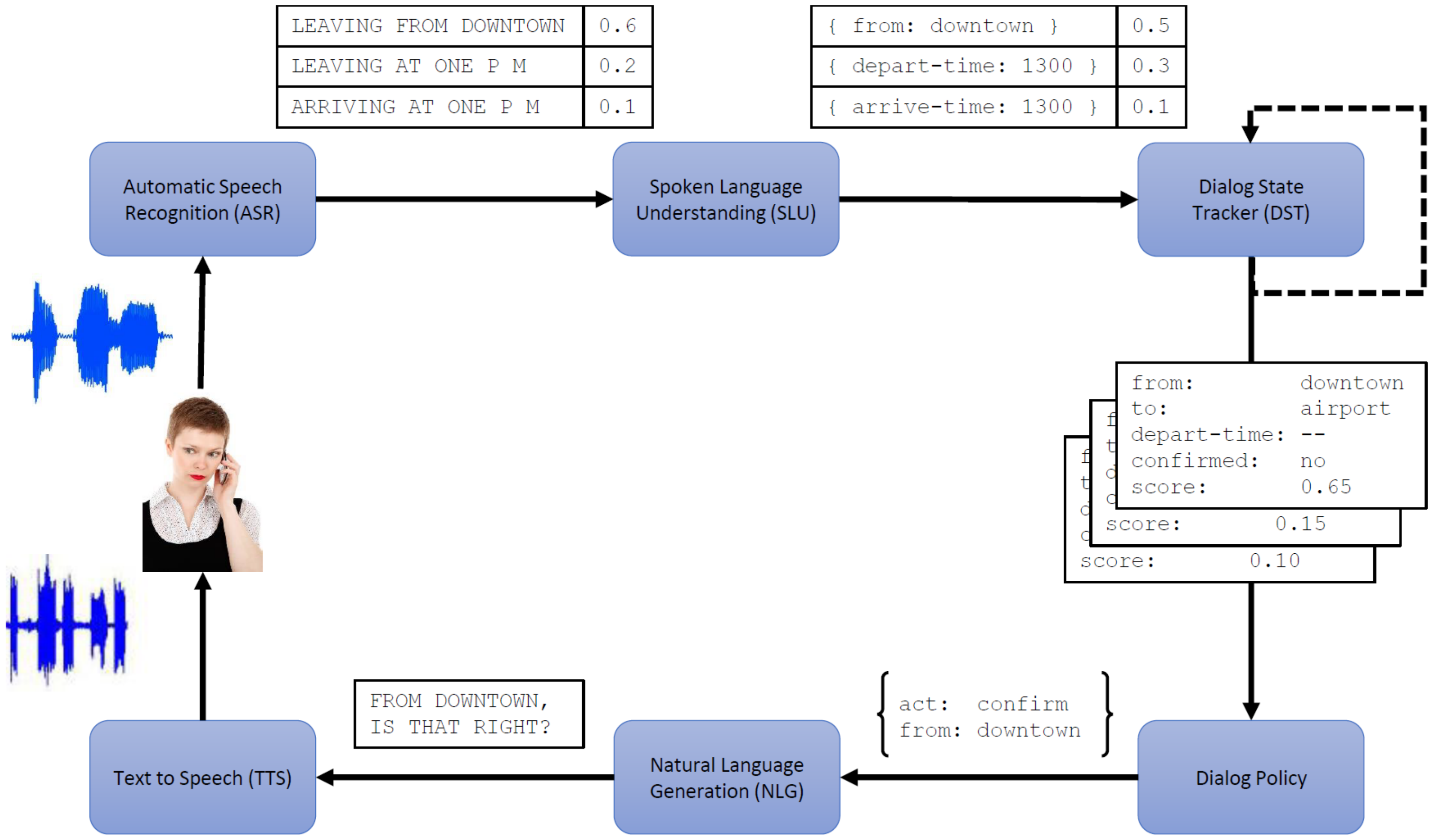
More sophisticated algorithm for slot filling: IOB Tagging

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

```
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
```

Modern Dialogue Systems - Dialogue-state Architecture



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
Text to Speech (TTS)
FROM DOWNTOWN, IS THAT RIGHT?
Natural Language Generation (NLG)
act: confirm
from: downtown
Dialog Policy

Modern Dialogue Systems - Dialogue-state Architecture

- NLU: like the GUS systems, the NLU component in extracts slot fillers from the user's utterance, but generally using machine learning rather than rules.
- Dialogue state tracker (DST) maintains the current state of the dialogue (which include the user's most recent dialogue act, plus the entire set of slot-filler constraints the user has expressed so far).
- The dialogue policy decides what the system should do or say next.
- Natural language generation (NGL) component.

References

- Speech and Language Processing (3rd ed. draft). Dan Jurafsky and James H. Martin
- http://deepdialogue.miulab.tw