# 15-494/694: Cognitive Robotics

## Dave Touretzky

Lecture 11:

Speech Generation and Recognition

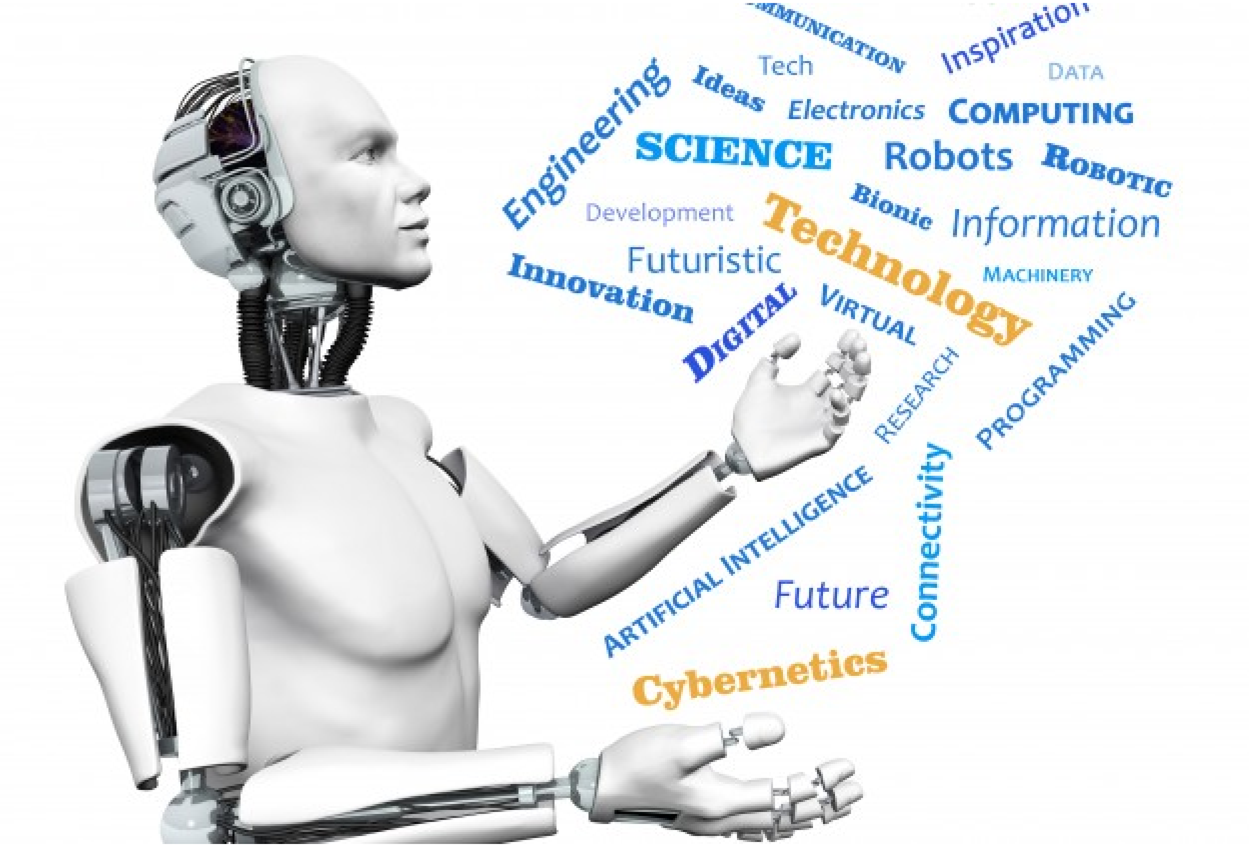


Image from http://www.futuristgerd.com/2015/09/10

# Speech Generation

- Cozmo does text-to-speech within the app
- Sends generated speech to the robot
- Parameters:
  - text
  - play_excited_animation [default False]
  - use_cozmo_voice [default True]
  - duration_scalar [default 1.0]
  - voice_pitch [-1.0 to 1.0]

# “Say” Node

- Constant case:

```
Say('hello there') =C=> next

Say('greetings', duration_scalar=0.5)
```

- Event-driven case:

```
Compute() =SayData=> Say() =C=> next
```

- Subclassing “Say”:

```
class SpeakBattery(Say)
```

# SpeakBattery

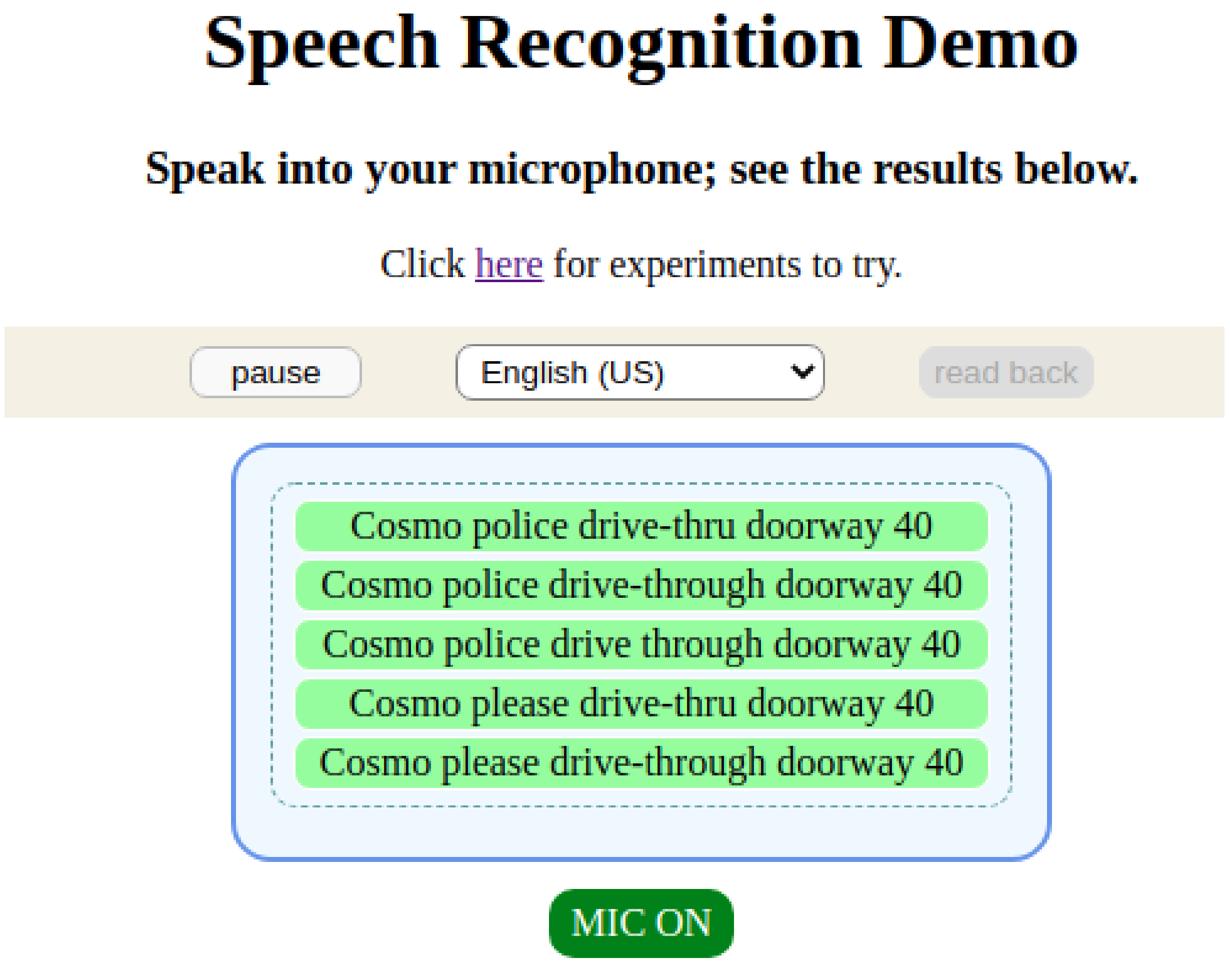
```
class SpeakBattery(Say):
  def start(self,event=None):
    self.text =
      f'battery voltage {robot.battery_voltage}'
    super().start(event)
```

# Speech Recognition

- Cozmo has no microphone
- Use the laptop's mic or a USB mic
- Recognition via the Google Speech API
  - Must have network access to function.
  - Biased towards conversational English, not arbitrary robot commands.
- “Cozmo grab cube1” heard as:
  - “cozmo **crab** cube1”
- “Cozmo please grab cube1” heard as:
  - “cozmo please **grab** cube1”

# Demo: Google Speech API

https://www.cs.cmu.edu/~dst/SpeechDemo



## Speech Recognition Demo

**Speak into your microphone; see the results below.**

Click here for experiments to try.

Cosmo police drive-thru doorway 40

Cosmo police drive-through doorway 40

Cosmo police drive through doorway 40

Cosmo please drive-thru doorway 40

Cosmo please drive-through doorway 40

# Requesting Speech Recognition

Speech recognition is turned off by default.
To turn it on: use speech=True in
StateMachineProgram.

```
class CozmoCommand(StateMachineProgram):
  def __init__(self):
    super().__init__(speech=True,
                     speech_debug=True)
```

# When To Listen

- Microphone is always on
- Use a wake word to indicate we're addressing the robot, or a cube.
  - “Cozmo, grab a cube”
  - “Cube1, turn green”
- You've seen this trick before:
  - “Alexa, ...”
  - “Hey Siri, ...”
  - “OK Google, ...”

# The =Hear()=> Transition

```
dispatch: Say('What now?')


dispatch =Hear('cozmo turn left')=>
  Turn(90) =C=> dispatch


dispatch =Hear('cozmo drive forward')=>
    Forward(50) =C=> dispatch
```

# String Matching

- Convert everything to lowercase
- Remove all punctuation
- Normalize homophones

# Homophones

- “Thesaurus” data structure defined in cozmo_fsm/speech.py
- Words:
  - cozmo ← cosmo, cosmos, cosimo, …
  - right ← write, wright
  - cube1 ← q1, coupon, cuban
- Phrases:
  - cube1 ← cube 1
  - paperclip ← paper clip

# Regular Expression Matching

- Uses the Python re package
- Example: optional words

  'cozmo ?(please|) drive forward'

- Be careful about spaces!
  - Example: scanning for keywords:

    'cozmo .* grab.*'

spaces on both sides of .* will be a problem
if the .* matches the null string

# Checking the Match Results

- When a =Hear=> transition fires, it offers a SpeechEvent to the target node(s).

- The SpeechEvent contains three items:
  - **string:** the string that was matched
  - **words:** list of words in the string
  - **result:** the match result from re.match
    - contains the groups defined by ( )

# Extracting Groups (1)

```
from cozmo_fsm import *


class Speech1(StateMachineProgram):
  def __init__(self):
    super().__init__(speech=True,
                     speech_debug=True)

  class Heard(Say):
    def start(self,event):
      obj = event.result.groups()[1]
      self.text = 'I will grab %s' % obj
      super().start(event)
```

# Extracting Groups (2)

```
$setup{
  loop: Say('what now')

  loop =Hear('cozmo ?(please|) grab
    (cube1|cube2|cube3)')=>
       self.Heard() =C=> loop

  loop =Hear=> Say('Pardon me?')
          =C=> loop
}
```

# Parsing

- We could write a parser for simple English commands and queries.
  - **Command:** “Cozmo, grab a cube”
  - **Command:** “Cozmo, find a door”
- This part is easy: each command directly translates to a state machine call.

# Queries

- Queries require access to a knowledge base and a mechanism for retrieval.

- What's in the knowledge base?
  - The world map
  - The robot's recent actions and plans
  - Recent object references
    - Necessary to resolve "it"

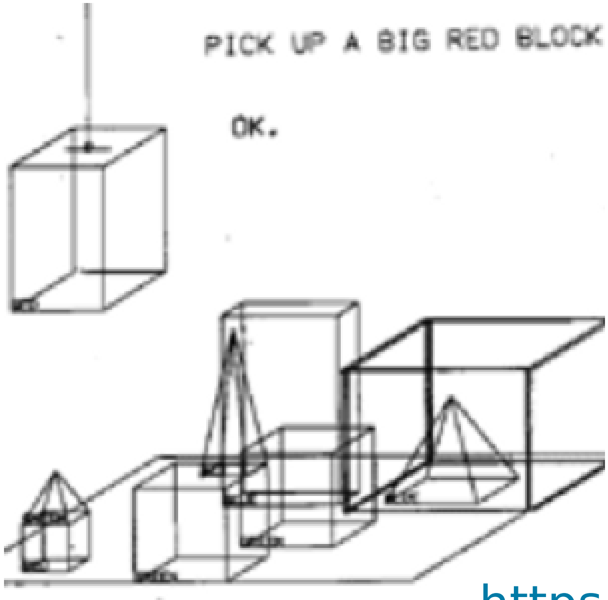
# Dialog

- “Cozmo, are there any red cubes?”
- “Cozmo, did you touch any green **ones**?”
- “Cozmo, where is the nearest door?”

- Pronoun resolution:
  - “Cozmo, have you seen a blue cube?”
  - “Where is **it**?”

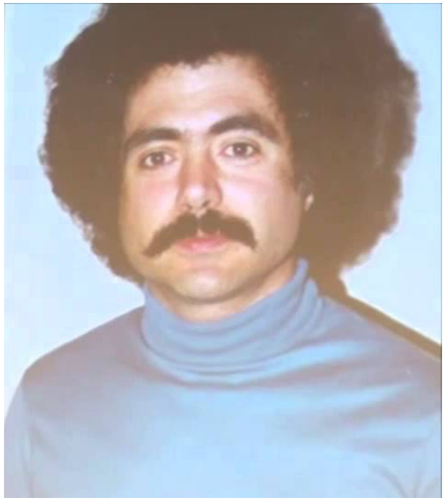
# Terry Winograd’s SHRDLU

- MIT PhD thesis, 1968–1970
- First natural language dialog system
- Robot arm in simulated blocks world.
- Teletype input, graphic display output.



PICK UP A BIG RED BLOCK

OK.



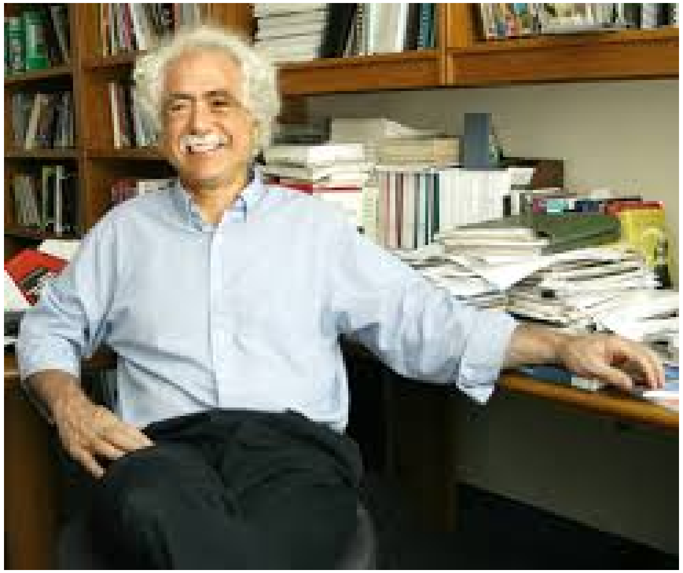
https://www.youtube.com/watch?v=bo4RvYJYOzI

# Sample SHRDLU Dialog

- What did the red cube support before you started to clean it off?
  - *The green pyramid.*
- There were five blocks to the left of the box then.
  - *No, only four of them: the red cube, two large green cubes and a large red block.*
- Put the blue pyramid on the block in the box.
  - *OK*

# Natural Language Today: Winograd Schema Challenge

- Two sentences differ only in a verb. Figure out the pronoun reference.
  - The city councilmen refused the demonstrators a permit because **they** [*feared / advocated* ] violence.



# GPT-4 From OpenAI

- GPT-4 is a large language model (LLM) with ~1.7 trillion parameters created by OpenAI.

- It is accessible via Python API calls.

- LLM's are a major advance in language understanding and reasoning.

- We will explore uses of GPT-4 with Cozmo.