Voice Dev 1

CS571: Building User Interfaces

Cole Nelson

Today's Warmup

- Clone today's code to your machine.
  - Run the command `npm install` inside of the `starter` and `solution` folders.
- Sign up for a wit.ai account.
  - This does *not* need to be your personal Facebook account, you can sign up for a Meta account using your @wisc.edu email!

Learning Objectives

1. Be able to define important conversational terms such as agent, utterance, and intent.
2. Be able to use JavaScript `async` / `await` syntax.
3. Be able to build a command-and-control chat agent!

Voice User Interfaces

VUIs are a common form of **agent-based design** as opposed to **direct manipulation**.

Conversational interfaces can be used to...

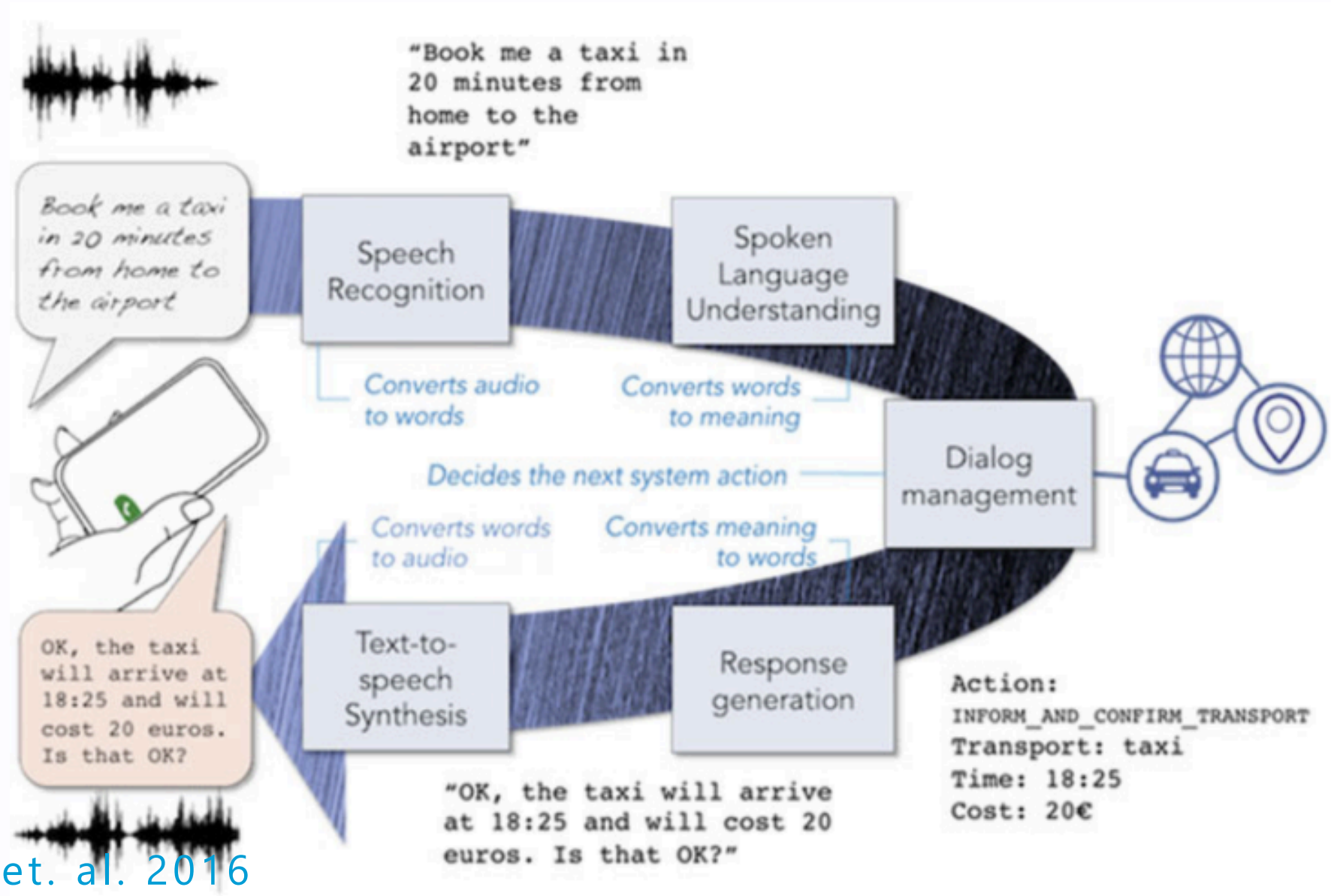
→ Address accessibility needs
→ Address context-specific problems (e.g. driving)
→ Augment the user experience

Voice User Interfaces

VUIs integrate a number of technologies and ideas...

1. Speech recognition
2. Spoken language understanding
3. Dialog management
4. Response generation
5. Text-to-speech synthesis

McTear et. al. 2016



McTear et. al. 2016

Order Domino's with Alexa!

Implementation Options

We focus on just one avenue of implementation!

- Wit.ai by Facebook
- DialogFlow by Google
- Watson Assistant by IBM
- Lex by Amazon
- Azure Bot Service by Microsoft

A Consideration: Killed by Google

Key Concepts in Wit.AI

- **Agent:** The overarching project consisting of *intents*, *utterances*, and *entities*.
- **Intents:** A higher level meaning of many *utterances*.
- **Utterances:** A string of words.

The goal of our **agent** is to extract the **intent** out of a new **utterance** and map it to a function.

Intents

Consider the following **utterances**...

- What is the weather like *tomorrow*?
- How's it looking out there *right now*?

What is the **intent** of these requests? They're both some sort of `weather_inquiry`!

These also have an **entity** of a time/date. We'll cover this next time.

Your Turn!

Let's create a Wit.AI comedian that can understand...

`why_chicken` e.g. "why did the chicken cross the road"

- We will reply with "to get to the other side!"

`tell_joke` e.g. "tell me a joke"

- We will reply with jokes fetched from the Jokes API!

Intents

```
"intents": [
    {
        "confidence": 0.992232100272184,
        "id": "2117974348567211",
        "name": "tell_joke"
    }
]
```

Each intent consists of a `confidence` from 0 to 1, as well as an `id` linked to the `name` of the intent.

The 0th intent is the best match.

`async` / `await`

How can we `return` from a `fetch`?

A Review of Async

Definition: not happening or done at the same time.

We `promise` something will happen in the future, `then` we do something.

`async` / `await` is just syntactic sugar for this!

Equivilant Handling!

```
function getLogos() {
  fetch("https://www.example.com/logos?amount=20")
  .then(res => res.json())
  .then((newLogos) => {
      setLogos(newLogos);
  })
}
```

```
async function getLogos() {
  const resp = await fetch("https://www.example.com/logos?amount=20");
  const newLogos = await resp.json();
  setLogos(newLogos);
}
```

Equivilant Handling - Errors

```
function getLogos() {
  fetch("https://www.example.com/logos?amount=20")
  .then(res => res.json())
  .then((newLogos) => {
      setLogos(newLogos);
  })
  .catch(e => alert("Something went wrong!"))
}
```

Equivilant Handling - Errors

```
async function getLogos() {
  try {
    const resp = await fetch("https://www.example.com/logos?amount=20");
    const newLogos = await resp.json();
    setLogos(newLogos);
  } catch (e) {
    alert("Something went wrong!")
  }
}
```

`async` / `await`

- equivalent way to handle asynchronous behavior
- `await` must be inside of an `async` function or at the top-level of the program
- `await` waits for right-hand-side to complete
- a synchronous function may spawn `async` behavior
- an `async` function always happens asynchronously, returning a `Promise`

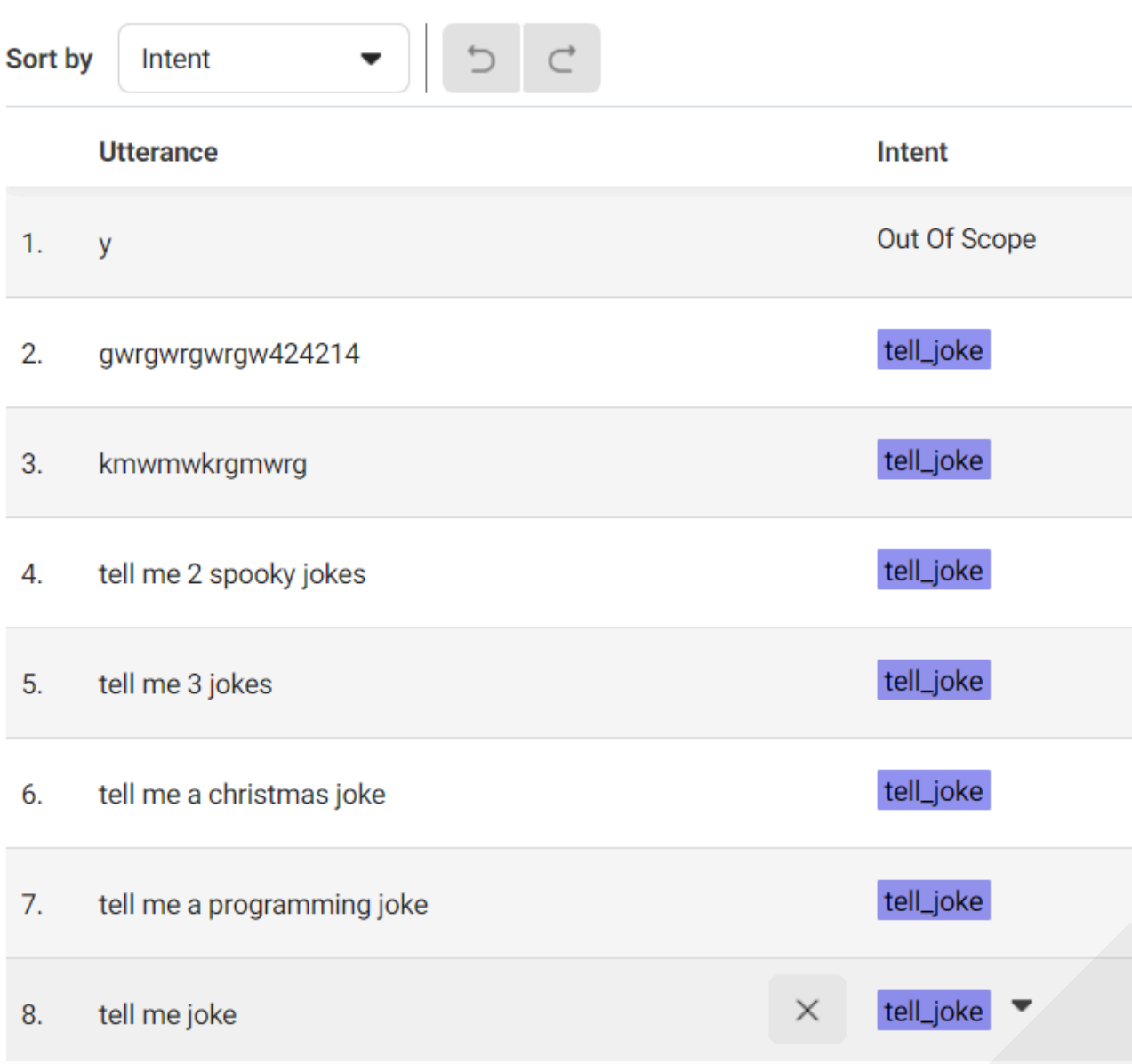
Your Turn!

Let's build a better agent using `async` / `await` ...

Reminder: AI!

There are no guarantees. Intent matching is still emerging technology.

Use as many utterances as you can!



Questions?