Virtual assistants and accessing data

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Virtual assistants

- Common chatbot use cases:
  - Scheduling a meeting
  - Booking a flight
  - Searching for a restaurant
- Require information about the outside world
- Need to interact with databases or APIs

Basic SQL

name	pricerange	area	rating
Bill's Burgers	hi	east	3
Moe's Plaice	low	north	3
Sushi Corner	mid	center	3

```
SELECT * from restaurants;
```

```
SELECT name, rating from restaurants;
```

```
SELECT name from restaurants
WHERE area = 'center' AND pricerange = 'hi';
```

SQLite with Python

```
import sqlite3
conn = sqlite3.connect('hotels.db')
c = conn.cursor()
c.execute("SELECT * FROM hotels WHERE area='south'
            and pricerange='hi'")
```

```
<sqlite3.Cursor at 0x10cd5a960>
```

```
c.fetchall()
```

```
[('Grand Hotel', 'hi', 'south', 5)]
```

SQL injection

```
# Bad Idea
query = "SELECT name from restaurant where area='{}'".format(area)
c.execute(query)
# Better
t = (area,price)
c.execute('SELECT * FROM hotels WHERE area=? and price=?', t)
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Exploring a DB with natural language

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

Example messages

- "Show me a great hotel"
- "I'm looking for a cheap hotel in the south of town"
- "Anywhere so long as it's central"

Parameters from text

```
message = "a cheap hotel in the north"
data = interpreter.parse(message)
data
```

```
{'entities': [{'end': '7', 'entity': 'price', 'start': 2, 'value': 'lo'},
  {'end': 26, 'entity': 'location', 'start': 21, 'value': 'north'}],
 'intent': {'confidence': 0.9, 'name': 'hotel_search'}}
```

```
params = {}
for ent in data["entities"]:
    params[ent["entity"]] = ent["value"]
params
```

```
{'location': 'north', 'price': 'lo'}
```

```
query = "select name FROM hotels"
filters = ["{}=?".format(k) for k in params.keys()]
filters
```

```
['price=?', 'location=?']
```

```
conditions = " and ".join(filters)
conditions
```

```
 'price=? and location=?'
```

```
final_q = " WHERE ".join([query, conditions])
final_q
```

```
'SELECT name FROM hotels WHERE price=? and location=?'
```

Responses

```
responses = [
        "I'm sorry :( I couldn't find anything like that",
        "what about {}?",
        "{} is one option, but I know others too :)"
        ]
results = c.fetchall()
len(results)
```

```
4
```

```
index = min(len(results), len(responses)-1)
responses[index]
```

```
'{} is one option, but I know others too :)'
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Incremental slot filling and negation

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Incremental filters

Basic memory

```
def respond(message, params):
    # update params with entities in message
    # run query
    # pick response
    return response, params

# initialise params
params = {}

# message comes in
response, params = respond(message, params)
```

Negation

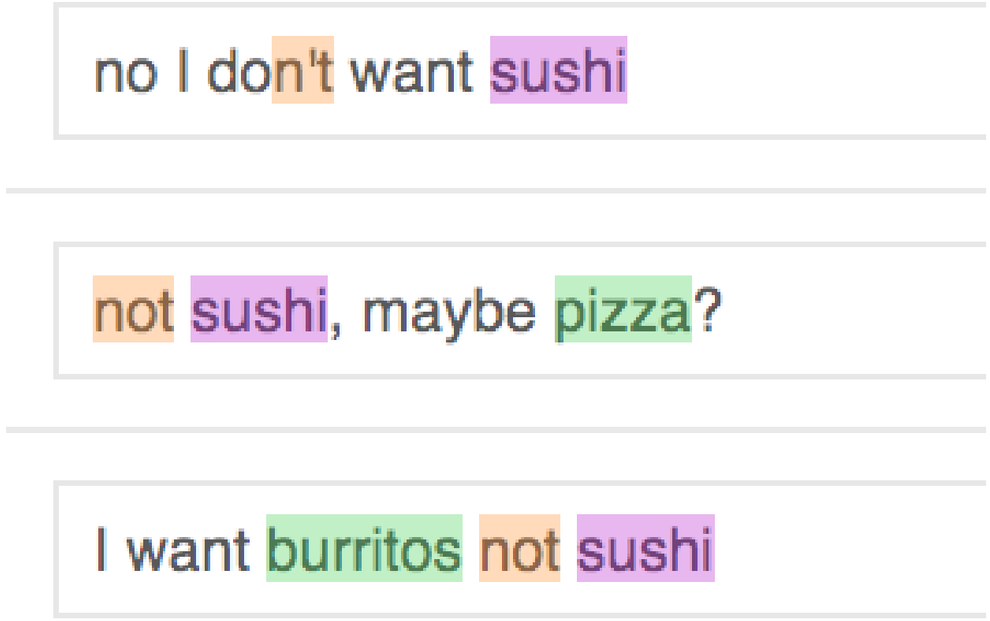
"where should I go for dinner?"

"no I don't like sushi"

"what about Sally's Sushi Place?"

"ok, what about Joe's Steakhouse?"

Negated entities



- assume that "not" or "n't" just before an entity means user wants to exclude this
- normal entities in green, negated entities in purple

Catching negations

```
doc = nlp('not sushi, maybe pizza?')
indices = [1, 4]
ents, negated_ents = [], []
start = 0
for i in indices:
     phrase = "{}".format(doc[start:i])
     if "not" in phrase or "n't" in phrase:
        negated_ents.append(doc[i])
     else:
         ents.append(doc[i])
     start = i
```

Let's practice!

BUILDING CHATBOTS IN PYTHON