



BUILDING CHATBOTS IN PYTHON

# Virtual assistants and accessing data

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

```
SELECT * from restaurants;
```

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

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

name	price	area	stars
Bill's Burgers	hi	east	3
Moe's Plaice	lo	north	3
Sushi Corner	mid	center	3



# SQLite with Python

```
In [1]: import sqlite3
```

```
In [2]: conn = sqlite3.connect('hotels.db')
```

```
In [3]: c = conn.cursor()
```

```
In [4]: c.execute("SELECT * FROM hotels WHERE area='south' and price='hi'")
```

```
Out[4]: <sqlite3.Cursor at 0x10cd5a960>
```

```
In [5]: c.fetchall()
```

```
Out[5]: [('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)
```



## BUILDING CHATBOTS IN PYTHON

**Let's practice!**



BUILDING CHATBOTS IN PYTHON

# Exploring a DB with natural language

Alan Nichol

Co-founder and CTO, Rasa



# 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

```
In [1]: message = "a cheap hotel in the north"
```

```
In [2]: data = interpreter.parse(message)
```

```
In [3]: data
```

```
Out[3]:
```

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

```
In [4]: params = {}
```

```
In [5]: for ent in data["entities"]:  
...:     params[ent["entity"]] = ent["value"]
```

```
In [6]: params
```

```
Out[6]: {'location': 'north', 'price': 'lo'}
```

# Creating a query from parameters

```
In [7]: query = "select name FROM hotels"
```

```
In [8]: filters = ["{}=?".format(k) for k in params.keys()]
```

```
In [9]: filters
```

```
Out[9]: ['price=?', 'location=?']
```

```
In [10]: conditions = " and ".join(filters)
```

```
In [11]: conditions
```

```
Out[11]: 'price=? and location=?'
```

```
In [12]: final_q = " WHERE ".join([query, conditions])
```

```
In [13]: final_q
```

```
Out[13]: 'SELECT name FROM hotels WHERE price=? and location=?'
```

# Responses

```
In [1]: responses = [  
        "I'm sorry :( I couldn't find anything like that",  
        "what about {}?",  
        "{} is one option, but I know others too :)"  
    ]
```

```
In [2]: results = c.fetchall()
```

```
In [3]: len(results)
```

```
Out[3]: 4
```

```
In [4]: index = min(len(results), len(responses)-1)
```

```
In [5]: responses[index]
```

```
Out[5]: '{} is one option, but I know others too :)'
```



## BUILDING CHATBOTS IN PYTHON

**Let's practice!**



BUILDING CHATBOTS IN PYTHON

# Incremental slot filling and negation

Alan Nichol

Co-founder and CTO, Rasa



# Incremental filters

I'm looking for a cheap  
hotel in the north of town

I'm sorry, I couldn't  
find anything like  
that.

what about mid range  
ones

Ann's BnB is a mid-  
priced hotel in the  
north of town

# Basic Memory

```
In [1]: def respond(message, params):
...:     # update params with entities in message
...:     # run query
...:     # pick response
...:     return response, params

# initialise params
In [2]: params = {}

# message comes in
In [3]: response, params = respond(message, params)
```



# Negation

"where should I go for dinner?"

"what about Sally's Sushi Place?"

"no I don't like sushi"

"ok, what about Joe's  
Steakhouse?"



# Negated entities

no I don't want sushi

not sushi, maybe pizza?

I want burritos not sushi

- 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

```
In [1]: doc = nlp('not sushi, maybe pizza?')

In [2]: indices = [1, 4]

In [3]: ents, negated_ents = [], []

In [4]: 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
```



## BUILDING CHATBOTS IN PYTHON

**Let's practice!**