Introduction to conversational software

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

A short history

Conversational software is not a new idea!

- Dates back to at least 1960s
- First wave: command line applications
- ELIZA: 1966

Course content

- Implementing smalltalk, ELIZA style
- How to use regex and ML to extract meaning from free-form text
- Build chatbots that can
  - Query a database
  - Plan a trip
  - Help you order coffee
- Handling statefulness

EchoBot I

```
USER: Hello!
```

```
BOT: I can hear you, you said: 'Hello!'
```

```
USER: How are you?
```

```
BOT: I can hear you, you said: 'How are you?'
```

EchoBot II

```
def respond(message):
    return "I can hear you! you said: {}".format(message)
def send_message(message):
    # calls respond() to get response
send_message("hello!")
```

```
USER: hello!
BOT : I can hear you! you said: hello!
```

EchoBot III

```
import time
time.sleep(0.5)
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Creating a personality

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Why personality?

- Difference between a command line app and a chatbot
- Makes chatbots and voice assistants more accessible and fun to use
- Your users will expect it!

Smalltalk

```
responses = {
    "what's your name?": "my name is EchoBot",
    "what's the weather today?": "it's sunny!"
}
def respond(message):
    if message in responses:
        return responses[message]
respond("what's your name?")
```

```
'my name is EchoBot'
```

Including variables

```
responses = {
    "what's today's weather?": "it's {} today"
}
weather_today = "cloudy"

def respond(message):
    if message in responses:
        return responses[message].format(weather_today)

respond("what's today's weather?")
```

```
"it's cloudy today"
```

Choosing responses

```
responses = {
    "what's your name?": [
      "my name is EchoBot",
      "they call me EchoBot",
      "the name's Bot, Echo Bot"
    ]
}
import random
def respond(message):
    if message in responses:
        return random.choice(responses[message])
respond("what's your name?")
```

```
"the name's Bot, Echo Bot"
```

Asking questions

```
responses = [ "tell me more!", "why do you think that?" ]

import random

def respond(message):
    return random.choice(responses)

respond("I think you're really great")
```

```
'why do you think that?'
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Text processing with regular expressions

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Regular expressions

- Match messages against known patterns
- Extract key phrases
- Transform sentences grammatically

The regex behind ELIZA

USER: *"do you remember when you ate strawberries in the garden?"*

ELIZA: *"How could I forget when I ate strawberries in the garden?"*

Pattern matching

```
import re
pattern = "do you remember .*"
message = "do you remember when you ate strawberries
            in the garden"
match = re.search(pattern, message)
if match:
    print("string matches!")
```

```
string matches!
```

Extracting key phrases

```
import re
pattern = "if (.*)"
message = "what would happen if bots took over the world"

match = re.search(pattern, message)
match.group(0)
```

```
'what would happen if bots took over the world'
```

```
match.group(1)
```

```
'bots took over the world'
```

Grammatical transformation

```
import re

def swap_pronouns(phrase):
    if 'I' in phrase:
        return re.sub('I', 'you', phrase)
    if 'my' in phrase:
        return re.sub('my', 'your', phrase)

    else:
        return phrase
swap_pronouns("I walk my dog")
```

```
'You walk your dog'
```

Putting it all together

```
pattern = 'do you remember (.*)'
message = 'do you remember when you ate strawberries
            in the garden'
phrase = pattern.search(pattern, message).group(1)
phrase
```

```
'when you ate strawberries in the garden'
```

```
response = choose_response(pattern)
response
```

```
'how could I forget {}'
```

Putting it all together

```
phrase = swap_pronouns(phrase)
phrase
```

```
'when I ate strawberries in the garden'
```

```
response.format(phrase)
```

```
'how could I forget when I ate strawberries in the garden'
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

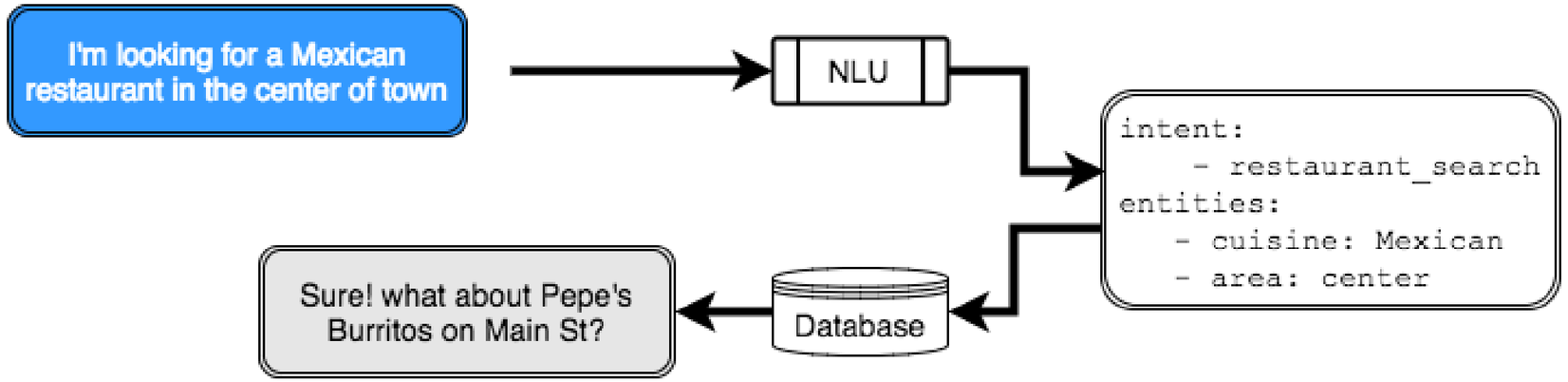
Understanding intents and entities

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

An example


I'm looking for a Mexican restaurant in the center of town
NLU
intent:
- restaurant_search
entities:
- cuisine: Mexican
- area: center
Database
Sure! what about Pepe's Burritos on Main St?

Intents

A `restaurant_search` can be expressed many different ways:

- I'm hungry
- Show me good pizza spots
- I want to take my boyfriend out for sushi
  - Can also be `request_booking`

Entities

Book a table for June 10th at a sushi restaurant in New York City

- NER = Named Entity Recognition

Regular expressions to recognize intents

- Simpler than machine learning approaches
- Highly computationally efficient
- Drawback:
  - Debugging regular expressions can become difficult

Using regular expressions

- `'|'` is equivalent to OR

```
re.search(r"(hello|hey|hi)", "hey there!") is not None
```

```
True
```

```
re.search(r"(hello|hey|hi)", "which one?") is not None
```

```
True
```

Using regular expressions

- `\b` matches the beginning or end of a word

```
re.search(r"\b(hello|hey|hi)\b", "hey there!") is not None
```

```
True
```

```
re.search(r"\b(hello|hey|hi)\b", "which one?") is not None
```

```
False
```

Using regex for entity recognition

```
pattern = re.compile('[A-Z]{1}[a-z]*')
message = """
Mary is a friend of mine,
she studied at Oxford and
now works at Google"""
pattern.findall(message)
```

```
['Mary', 'Oxford', 'Google']
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Word vectors

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

Machine learning

- Programs which can get better at a task by being exposed to more data
- Identifying which intent a user message belongs to

Vector representations

"can you help me please?"

Units	examples	vectors
characters	`"c", "a", "n",...`	`v_c, v_a, v_n, ...`
words	`"can", "you", ...`	`v_{can}, v_{you}, ...`
sentences	`"can you help..."`	`v_{can you help ...}`

Word vectors

Context	Candidates
let's meet at the ___ tomorrow	office, gym, park, beach, party
I love going to the ___ to play with the dogs	beach, park

- Word vectors try to represent *meaning* of words
- Words which appear in similar context have similar vectors

Word vectors are computationally intensive

- Training word vectors requires a lot of data
- High quality word vectors are available for anyone to use
- GloVe algorithm
  - Cousin of word2vec
- spaCy

```
import spacy
nlp = spacy.load('en')
nlp.vocab.vectors_length
```

```
300
```

```
doc = nlp('hello can you help me?')
for token in doc:
    print("{} : {}".format(token, token.vector[:3]))
```

```
hello : [ 0.25233001  0.10176    -0.67484999]
can : [-0.23857     0.35457    -0.30219001]
you : [-0.11076     0.30785999 -0.51980001]
help : [-0.29370001  0.32253    -0.44779   ]
me : [-0.15396     0.31894001 -0.54887998]
? : [-0.086864    0.19160999  0.10915   ]
```

Similarity

- Direction of vectors matters
- "Distance" between words = angle between the vectors
- Cosine similarity
  - 1: If vectors point in the same direction
  - 0: If they are perpendicular
  - -1: If they point in opposite directions

.similarity()

- "can" and "cat" are spelled similarly but have low similarity
- but "cat" and "dog" have high similarity

```
import spacy
nlp = spacy.load('en')
doc = nlp("cat")
doc.similarity(nlp("can"))
```

```
0.30165292161215396
```

```
doc.similarity(nlp("dog"))
```

```
0.80168555173294953
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Intents and classification

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Supervised learning

- A classifier predicts the intent label given a sentence
- "Fit" classifier by tuning it on *training data*
- Evaluate performance on *test data*
- Accuracy: the fraction of labels we predict correctly

ATIS dataset

- Thousands of sentences with labeled intents and entities
- Collected from a real flight booking service
- Intents like
  - `atis_flight`
  - `atis_airfare`

ATIS dataset II

```
sentences_train[:2]
labels_train[:2]
```

```
[ "atis_flight",
  "atis_flight"]
```

```
["i want to fly from boston at
  838 am and arrive in denver at
  1110 in the morning",
  "what flights are available
  from pittsburgh to baltimore
  on thursday morning"]
```

```
import numpy as np
X_train_shape = (
    len(sentences_train),
    nlp.vocab.vectors_length)
X_train = np.zeros(X_train_shape)
for sentence in sentences_train:
    X_train[i,:] = nlp(sentence).vector
```

Nearest neighbor classification

- Need training data
  - Sentences which we've already labeled with their intents
- Simplest solution:
  - Look for the labeled example that's most similar
  - Use its intent as a best guess
- Nearest neighbor classification

Nearest neighbor classification in scikit-learn

```
from sklearn.metrics.pairwise import cosine_similarity
test_message = """
i would like to find a flight from charlotte
to las vegas that makes a stop in st. louis"""
test_x = nlp(test_message).vector
scores = [
    cosine_similarity(X[i,:], test_x)
    for i in range(len(sentences_train)
]
labels_train[np.argmax(scores)]
```

```
'atis_flight'
```

Support vector machines

- Nearest neighbors is very simple - we can do better
- SVM / SVC: support vector machine / classifier

```
from sklearn.svm import SVC
clf = SVC()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Entity extraction

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

Beyond keywords: context

play Jailhouse Rock by Elvis

- Keywords don't work for entities you haven't seen before
- Use contextual clues:
  - Spelling
  - Capitalization
  - Words occurring before & after
- Pattern recognition

Pre-built Named Entity Recognition

```
import spacy
nlp = spacy.load('en')
doc = nlp("my friend Mary has worked at Google since 2009")
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Mary PERSON
Google ORG
2009 DATE
```

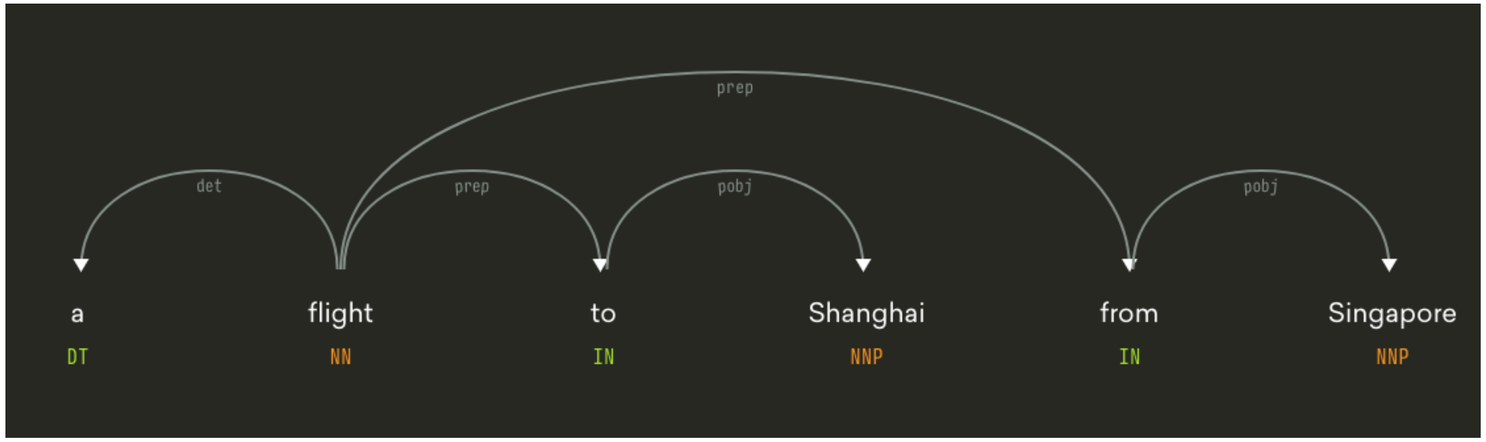
Roles

I want a flight from Tel Aviv to Bucharest

show me flights to Shanghai from Singapore

```
pattern_1 = re.compile('.* from (.*) to (.*)')

pattern_2 = re.compile('.* to (.*) from (.*)')
```




```
doc = nlp('a flight to Shanghai from Singapore')
shanghai, singapore = doc[3], doc[5]
list(shanghai.ancestors)
```

```
[to, flight]
```

```
list(singapore.ancestors)
```

```
[from, flight]
```

Shopping example

```
doc = nlp("let's see that jacket in red and some blue jeans")
items = [doc[4], doc[10]]  # [jacket, jeans]

colors = [doc[6], doc[9]]  # [red, blue]
for color in colors:
    for tok in color.ancestors:
        if tok in items:
            print("color {} belongs to item {}".format(color, tok))
            break
```

```
color red belongs to item jacket
color blue belongs to item jeans
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Robust NLU with Rasa

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

Rasa NLU

- Library for intent recognition & entity extraction
- Based on spaCy, scikit-learn, & other libraries
- Built in support for chatbot specific tasks

Rasa data format

```
from rasa_nlu.converters import load_data
training_data = load_data("./training_data.json")
import json
print(json.dumps(data.training_examples[22], indent=2))
```

```
{ "text": "i'm looking for a place in the north of town",
  "intent": "restaurant_search",
  "entities": [
    { "start": 31,
      "end": 36,
      "value": "north",
      "entity": "location" }
  ]
}
```

Interpreters

```
message = "I want to book a flight to London"
interpreter.parse(message))
```

```
{ "intent": {
    "name": "flight_search",
    "confidence": 0.9
  },
  "entities": [
    { "entity": "location",
      "value": "London",
      "start": 27,
      "end": 33
    }
  ]
}
```

Rasa usage

```
# Creating a model
from rasa_nlu.config import RasaNLUConfig
from rasa_nlu.model import Trainer
config = RasaNLUConfig(
            cmdline_args={"pipeline": "spacy_sklearn"})
trainer = Trainer(config)
interpreter = trainer.train(training_data)
```

Rasa pipelines

```
spacy_sklearn_pipeline = [
  "nlp_spacy",
  "ner_crf",
  "ner_synonyms",
  "intent_featurizer_spacy",
  "intent_classifier_sklearn"  ]
# These two statements are identical:
RasaNLUConfig(cmdline_args={"pipeline": spacy_sklearn_pipeline})
```

```
<rasa_nlu.config.RasaNLUConfig at 0x10f60aa90>
```

```
RasaNLUConfig(cmdline_args={"pipeline": "spacy_sklearn"})
```

```
<rasa_nlu.config.RasaNLUConfig at 0x10f60aa20>
```

Conditional random fields

- Machine Learning model, popular for named entity recognition
  - can perform well even with small training data

Handling typos

round trip fares from baltimore to philadelphia under 1000 dollas

please show me airlines with fligths from philadelphia to dallas

```
pipeline = [
    "nlp_spacy",
    "intent_featurizer_spacy",
    "intent_featurizer_ngrams",
    "intent_classifier_sklearn"
]
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Virtual assistants and accessing data

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

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

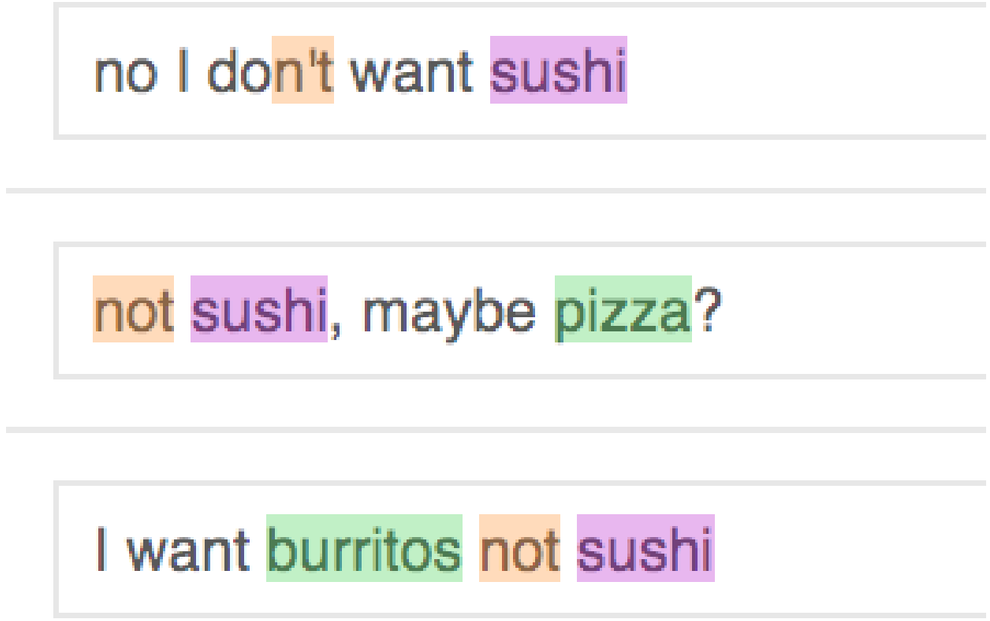
"where should I go for dinner?"

"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

Stateful bots

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

What do we mean by stateful?

"I love stateless systems!"

"don't they have drawbacks?"

"don't *what* have drawbacks?"

State machines

- Browsing
- Providing address, billing info
- Order complete

Implementing a state machine

```
INIT = 0
CHOOSE_COFFEE = 1
ORDERED = 2
```

Example rules:

```
policy_rules = {
    (INIT, "order"): (CHOOSE_COFFEE, "ok, Columbian or Kenyan?"),
    (CHOOSE_COFFEE, "specify_coffee"):
    (ORDERED, "perfect, the beans are on their way!"),
}
```

Using the state machine

```
state = INIT
def respond(state, message):
    (new_state, response) = policy_rules[(state,
                                          interpret(message))]
    return new_state, response

def send_message(state, message):
    new_state, response = respond(state, message)
    return new_state

state = send_message(state, message)
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Asking questions & queuing answers

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Reusable patterns

"I'd like some Kenyan beans"

"I'm sorry, we're out of those. Shall I order some Brazilian ones for you?"

"Yes please"

"Can I get a box of 200 brown filters"

"I'm sorry, we're out of those, but I can get your some white ones. Should I order those for you?"

"Yes please"

Pending actions

- Policy returns two values: Selected `action` and `pending_action`
- `pending_action` is saved in the outer scope
- If we get a "yes" intent and there is a pending action, we execute it
- If we get a "no" intent, we wipe any pending actions

"I'd like to order some coffee"

```
state = INIT
action = "request_auth"
pending_state = AUTHED
```

- Sounds good! I'd love to help you but you'll have to log in first, what's your phone number?

"555-12345"

```
state = AUTHED
action = "acknowledge_auth"
pending_state = None
```

- Perfect! welcome back :)

Let's practice!

BUILDING CHATBOTS IN PYTHON

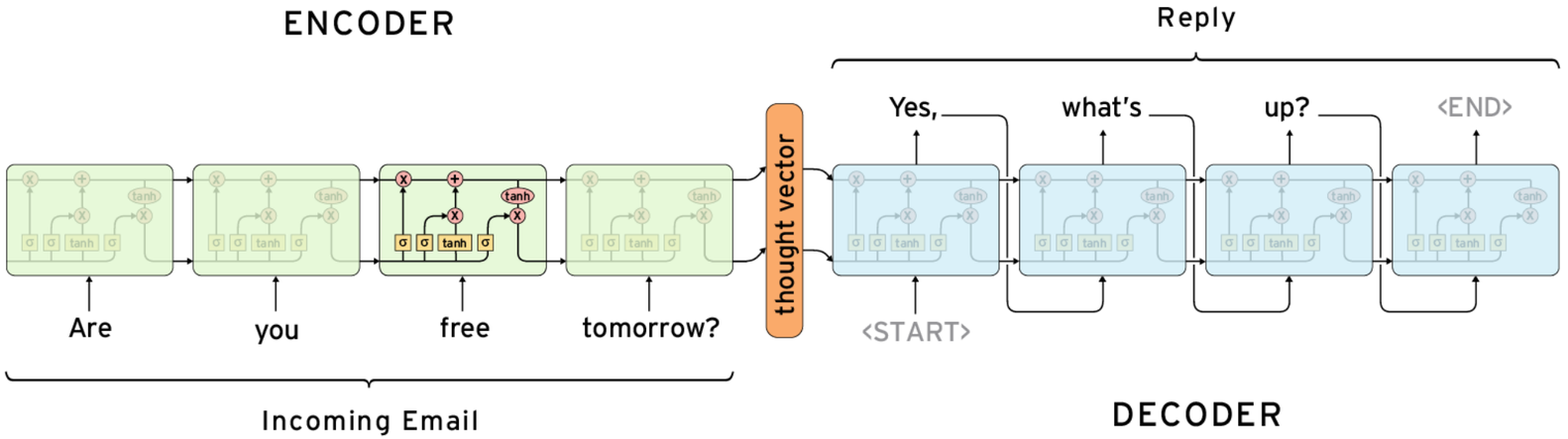
Frontiers of dialogue technology

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

datacamp

A neural conversational model



"What do you think of Cleopatra?" "Oh, she's very regal"

"What do you think of Messi?" "He's a great player"

Seq2seq

- Machine translation
- Completely data driven, no hand-crafting
- Requires large amount of data
- No guarantee that output is coherent
- Difficult to integrate DB / API calls & other logic

Grounded dialogue systems

- Systems you've built in this course: hand-crafted
- Seq2seq: Data driven
- ML based dialogue systems:
  - NLU
  - Dialogue state manager
  - API logic
  - Natural language response generator
- Human pretend to be a bot: "Wizard of Oz" technique
- Reinforcement learning
  - Receives a reward for a successful conversation

Language generation

- Not recommended if building a bot
- Pre-trained neural network which can generate text
- Scripts of every episode of The Simpsons

Generating sample text

```
generated = sample_text(
    saved_params,
    temperature,
    num_letters=num_letters,
    init_text=text
)
```

Let's practice!

BUILDING CHATBOTS IN PYTHON

Congratulations!

BUILDING CHATBOTS IN PYTHON

Alan Nichol
Co-founder and CTO, Rasa

Let's practice!

BUILDING CHATBOTS IN PYTHON