BLS, Integrations, DialogFlow, and Rasa (part 2)

Lecture 14

Review: Tools and Web Services

- A **Web service** is a **program** that runs on a **server** that interacts with other computers and software over the **Internet**
  - For your prototype front- and back-ends, consider `screen` or `tmux`
    - Run services that persist without constant ssh’ing
  - Web services interact using serialized data (**JSON** payloads)
    - Basically, giant “standardized” strings that contain relevant information
    - Historically, XML or per-application custom protocols have been used
      - e.g., HTTP (still used today) is simply customized strings
        - GET google.com/
        - Response 200, 404, 403, 500, etc.
- Front end → JSON → CAI platform → JSON → BLS

One-Slide Summary: DF/Rasa/Integrations

- **Conversational AI platforms** generally are **web-based** tools that provide a way to design:
  - **Intents** for classifying text
  - **Slot extraction** (slot-value pairing, entity extraction) for pulling semantics
  - **Dialogue tracking** (states, contexts) for handling multiple "turns"
- **DialogFlow** is a Google platform
  - Design intents with contexts, use WebHook for "fulfillment" of business logic
- **Rasa** is an open-source set of command-line tools
  - Design intents and "stories", use Actions for business logic

Comparing Platforms

- Large software systems come with various tradeoffs

- For Conversational AI, platforms balance
    - Speed of development
    - Ease of experience design
    - Testing
    - Training time
    - End-user experience

Clinc

- You design a **state graph** that models flow through a conversation
  - A state encompasses an **intent**
  - Intent classification model is trained via examples per transition between states
- The platform is reached remotely via the /v1/query API endpoint
  - Your front-end uses this API (e.g., website, Alex, etc.)
- The platform engages your BLS on a per-utterance basis
  - Move through the state graph on each utterance

Clinc: State Graph and Competencies

- The magic sauce to Clinc is the **State Graph** and notion of **competencies**

- These abstractions are meant to ease **development**
  - HITR: A human that helps you with something is *competent* at a few skills
    - A bank teller *knows* how to open an account
    - A travel agent *knows* how to get good airline tickets
    - A real estate agent (hopefully) *knows* how to not sell you a broken garbage house

TOM HANKS
SHELLEY LONG
For everyone who's ever been deeply in Love or deeply in Debt.
SOLD
THE MONEY PIT

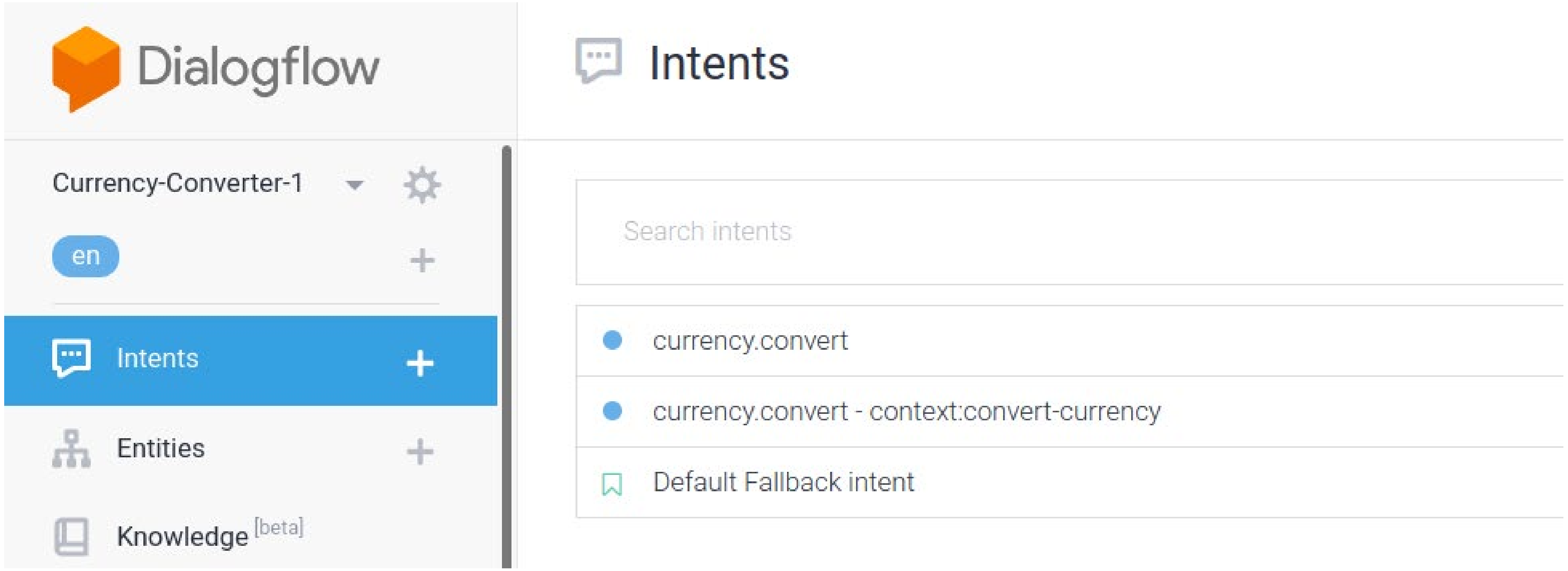
Clinc: Benefits and Drawbacks

- **Benefit:**
  - You design an experience around a state graph (complexity management)
  - Integration of crowdsourcing (simpler data collection and curation)

- **Drawback**:
  - Training is complicated
    - Example overlap
    - Single intent in multiple competencies
    - Stateful classifiers increase training time

Google DialogFlow

- DialogFlow allows specifying a list of **intents**



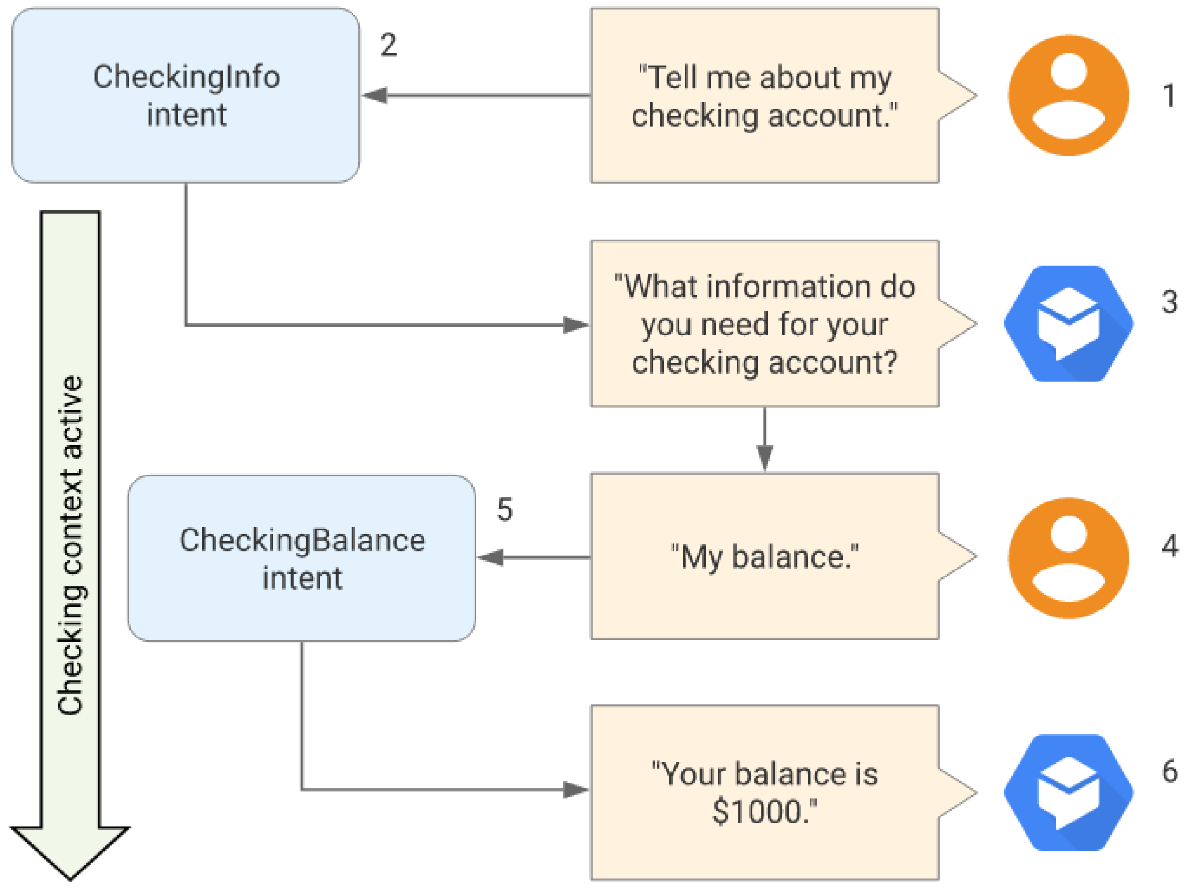
currency.convert

currency.convert - context:convert-currency

Default Fallback intent

Google DialogFlow

- No formal notion of *state*; instead, DialogFlow uses **contexts**



DialogFlow: Context

- Similar to a Clinc **competency**, a DialogFlow **context** is a set of intents that share **slots (entities/parameters)**
- You can design responses and intents knowing that you can extract and fill slots using contexts
- Like with an intent or state, multiple **contexts** can become *active* when an intent is matched

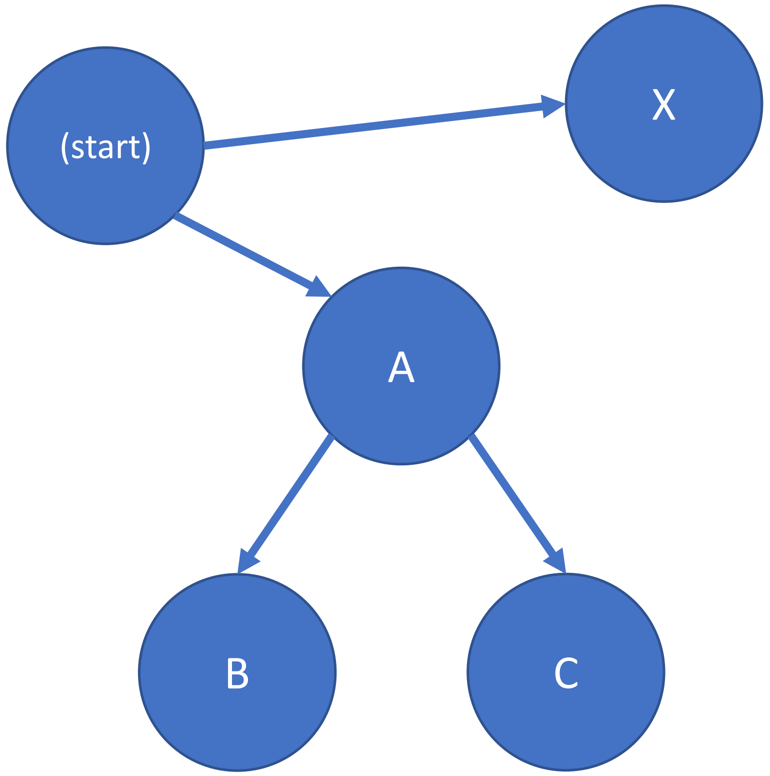
DialogFlow Contexts

- Contexts bias intent classification
  - Input contexts: helps to decide between potentially-ambiguous intents
    - "car" vs. "truck" in a vehicle-shopping bot
  - Output contexts: When an intent is classified, assign set of output contexts
    - "select_vehicle" in a vehicle-shopping bot

- Contexts also support timing
  - Default 20 minutes
  - 5 conversational turns

DialogFlow: Follow-up intents

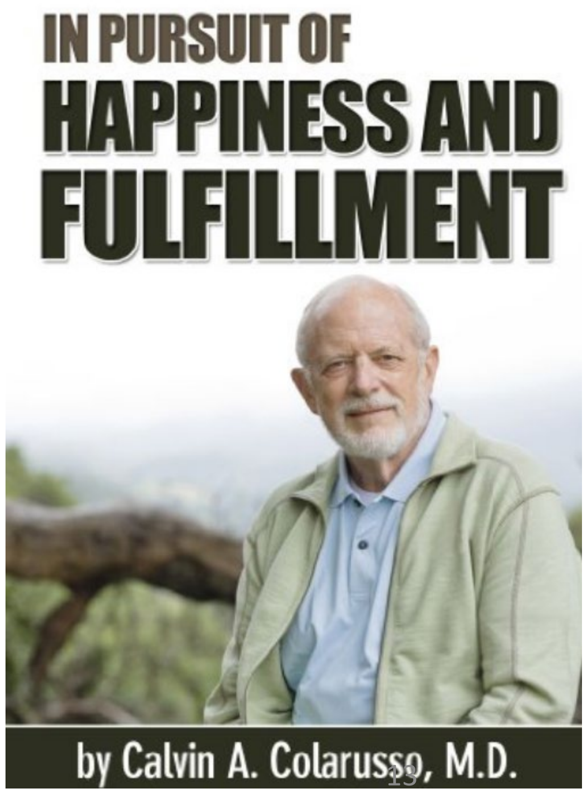
- Within an intent, a **follow-up** allows intent classification having already classified an intent
    - Helps maintain statefulness
    - A: I like candy
        - Yes:
            - B: Do you want chocolate?
        - No:
            - C: Do you want asparagus?
    - X: I like turtles



B and C are follow-ups to A

DialogFlow: Fulfillment

- (same as Business Logic)
- Basically, you run a web service that "fulfills" a request
- Set contexts, set intents, set responses, etc.
- You can also create fulfillment "Functions" that are integrated into DialogFlow

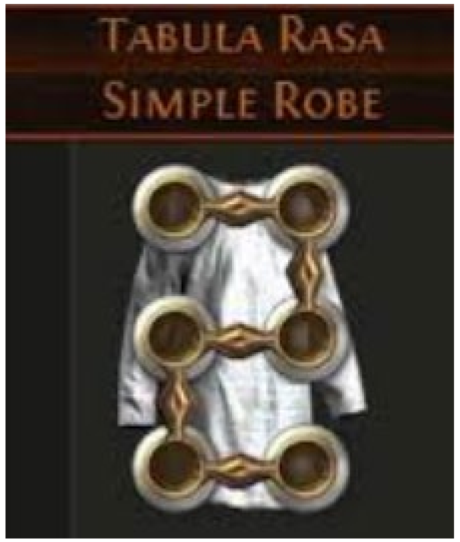


DialogFlow: Benefits and Drawbacks

- **Benefits:**
  - Massive Google resources and availability
  - List of intents may be “simpler” than a state graph
  - Built-in support for BLS (i.e., integrated “Fulfillment”)

- **Drawbacks:**
  - Intents, contexts, and follow-ups may introduce design complexity
  - Limitations of contexts and follow-ups promote serial dialogues
  - Cloud deployment?
    - Where is the data stored?

Rasa



- Open source Conversational AI platform

- Lots of plaintext, command-line interactions

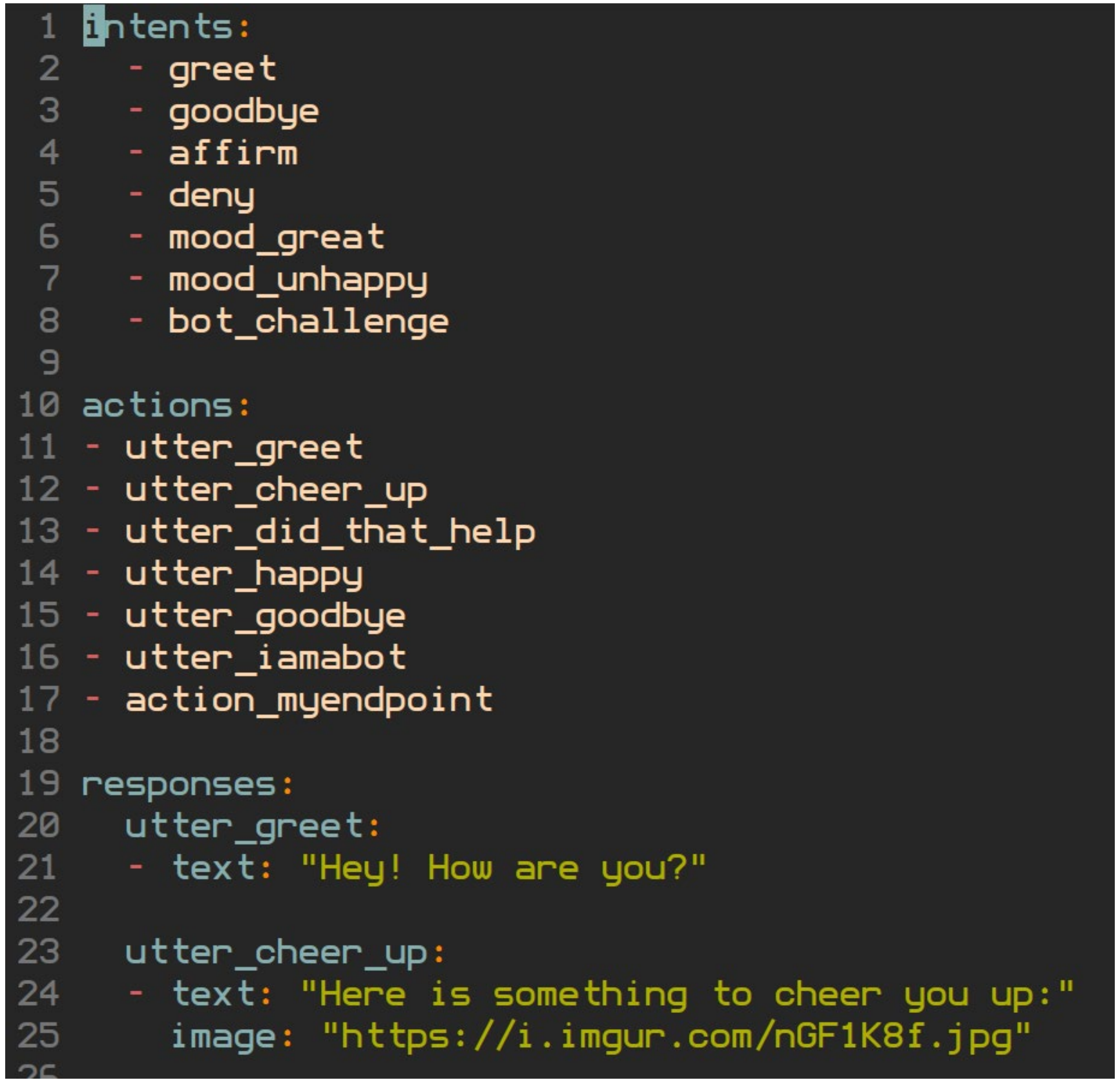
- Intents and slots as expected
  - Define a list of intents and corresponding examples
  - Label a few example slots within each example

- Rasa runs as a background service that you run locally

Rasa: Domains

- A Conversational AI built in Rasa is defined by a **domain**
  - Provides a notion of "scope" for what the virtual assistant does

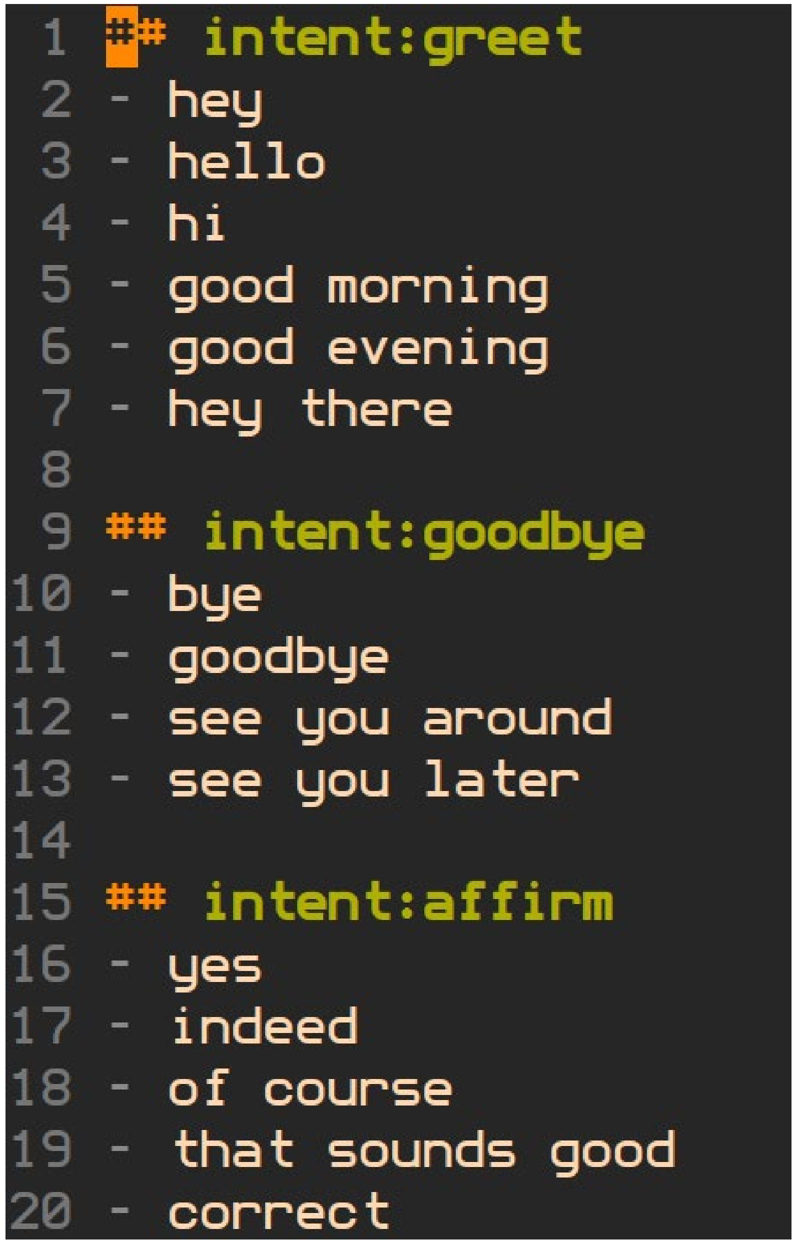
- The domain lists out all the intents, slots, responses



```
intents:
  - greet
  - goodbye
  - affirm
  - deny
  - mood_great
  - mood_unhappy
  - bot_challenge

actions:
- utter_greet
- utter_cheer_up
- utter_did_that_help
- utter_happy
- utter_goodbye
- utter_iamabot
- action_myendpoint

responses:
  utter_greet:
  - text: "Hey! How are you?"

  utter_cheer_up:
  - text: "Here is something to cheer you up:"
    image: "https://i.imgur.com/nGF1K8f.jpg"
```

Rasa: Intents

- Defined in MarkDown files
- Just a big list of intent names and examples

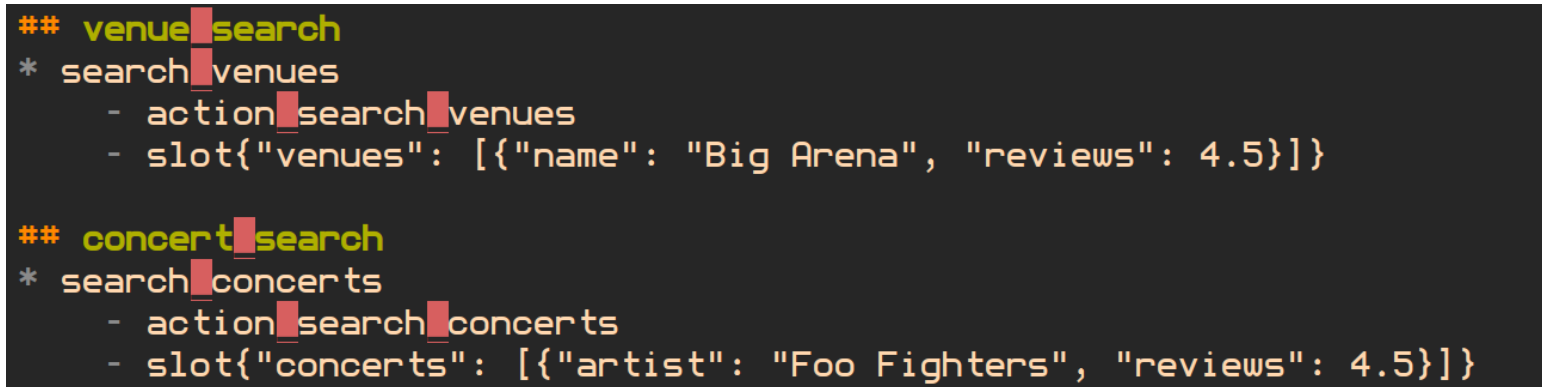


```
## intent:greet
- hey
- hello
- hi
- good morning
- good evening
- hey there

## intent:goodbye
- bye
- goodbye
- see you around
- see you later

## intent:affirm
- yes
- indeed
- of course
- that sounds good
- correct
```

Rasa: Slots

- Defined in domain, provided in examples

- Rasa supports **types** of slots
  - List
  - String
  - Float
  - Range
  - Enumerated



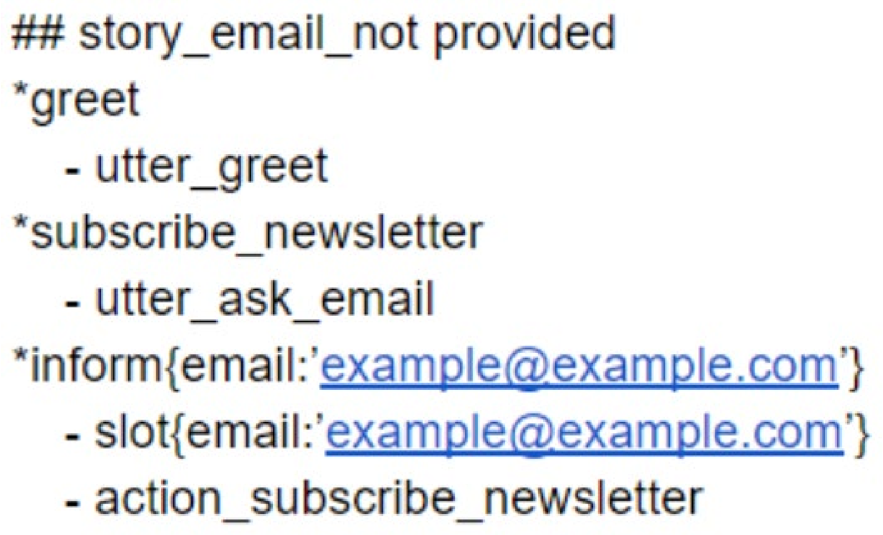
```
## venue_search
* search_venues
    - action_search_venues
    - slot{"venues": [{"name": "Big Arena", "reviews": 4.5}]}

## concert_search
* search_concerts
    - action_search_concerts
    - slot{"concerts": [{"artist": "Foo Fighters", "reviews": 4.5}]}
```

- Rasa also works with a variety of channels
  - (slots can be strange... e.g., JSON objects)

Rasa: Dialogue tracking with Stories

- Instead of *states*, Rasa uses **stories**

- You provide examples of sequences of intents and actions to take
  - Rasa *learns* sequences of intents to classify based on those examples



```
## story_email_not provided
*greet
   - utter_greet
*subscribe_newsletter
   - utter_ask_email
*inform{email:'example@example.com'}
   - slot{email:'example@example.com'}
   - action_subscribe_newsletter
```

Rasa: Actions

- (same as business logic and fulfillment)

- **Actions** are a kind of special "intent" that can get classified
  - The virtual assistant can predict going to perform an action
  - You indicate Actions in Stories

- You can write a web service that is engaged
  - You need an "action" in your user stories

Rasa: Benefits and Drawbacks

- **Benefits:**
  - Command-line and plaintext
  - Example-based input allows coercing statefulness
  - Built-in handling of “actions” (very Unix-y)

- **Drawbacks:**
  - Open-source project
    - Ever tried OpenOffice?
  - Lack of designed statefulness entails complex examples

OPEN SOURCE DEVELOPERS
What my friends think I do
What my spouse thinks I do
What the users think I should do
What enterprise companies think I do
What I think I do
What I actually do

Conversational Platforms: Summary

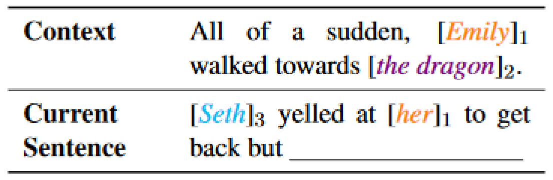
- These platforms provide coarse abstractions for modeling conversations
- They *assume* conversations take place turn-at-a-time
  - Intent-based
  - Slots to extract
  - Etc.
- The developer still needs to:
  - Model the conversation
  - Collect and curate the training data
  - Write code to do real stuff (i.e., front end and business logic)

Response Generation

- Conversational AI platforms can figure out what a user is talking about
- However, AI is still far too limited to actually "think"
  - Need human designer to figure out how to **take action** and **form responses**
- This manifests in two ways
  - Business Logic / Webhook / Action: Code you write to actually do stuff
  - **Response Generation:** filling out a response that an end-user will understand

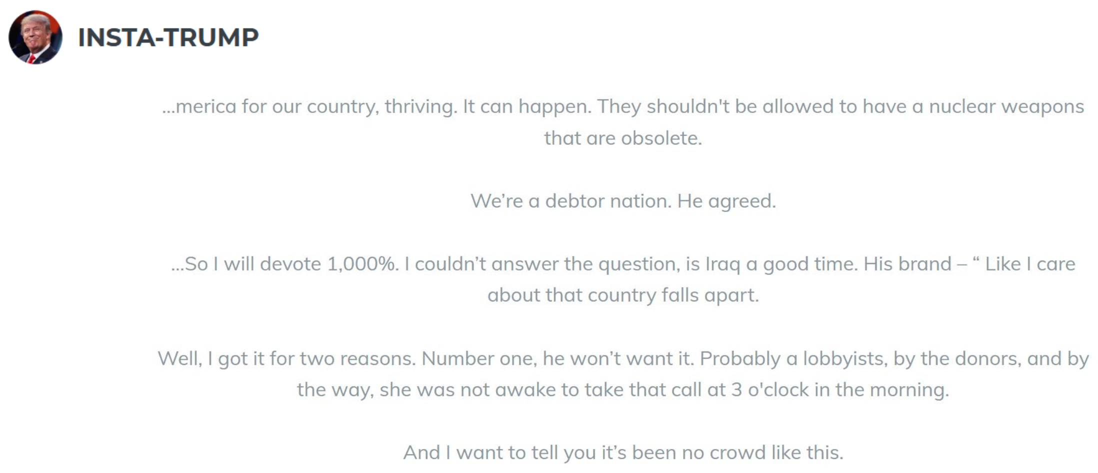
Response Generation

- This is ***not*** the same as **Natural Language Generation**



Context	All of a sudden, [*Emily*]$_1$ walked towards [*the dragon*]$_2$.
Current Sentence	[*Seth*]$_3$ yelled at [*her*]$_1$ to get back but ____________

Figure 1: An example of entity-labeled story data. The brackets indicate which words are part of entity mentions. Mentions marked with the same number refer to the same entity. The goal is to continue the story in a coherent way. The actual story reads, "*Seth yelled at her to get back but she ignored him.*"



INSTA-TRUMP

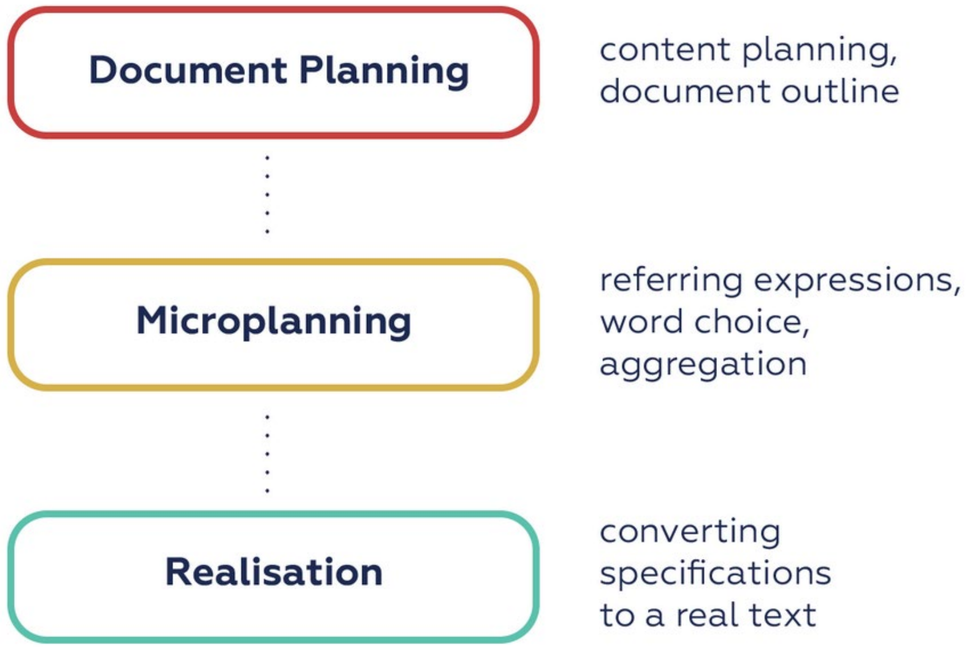
...merica for our country, thriving. It can happen. They shouldn't be allowed to have a nuclear weapons that are obsolete.

We're a debtor nation. He agreed.

...So I will devote 1,000%. I couldn't answer the question, is Iraq a good time. His brand – " Like I care about that country falls apart.

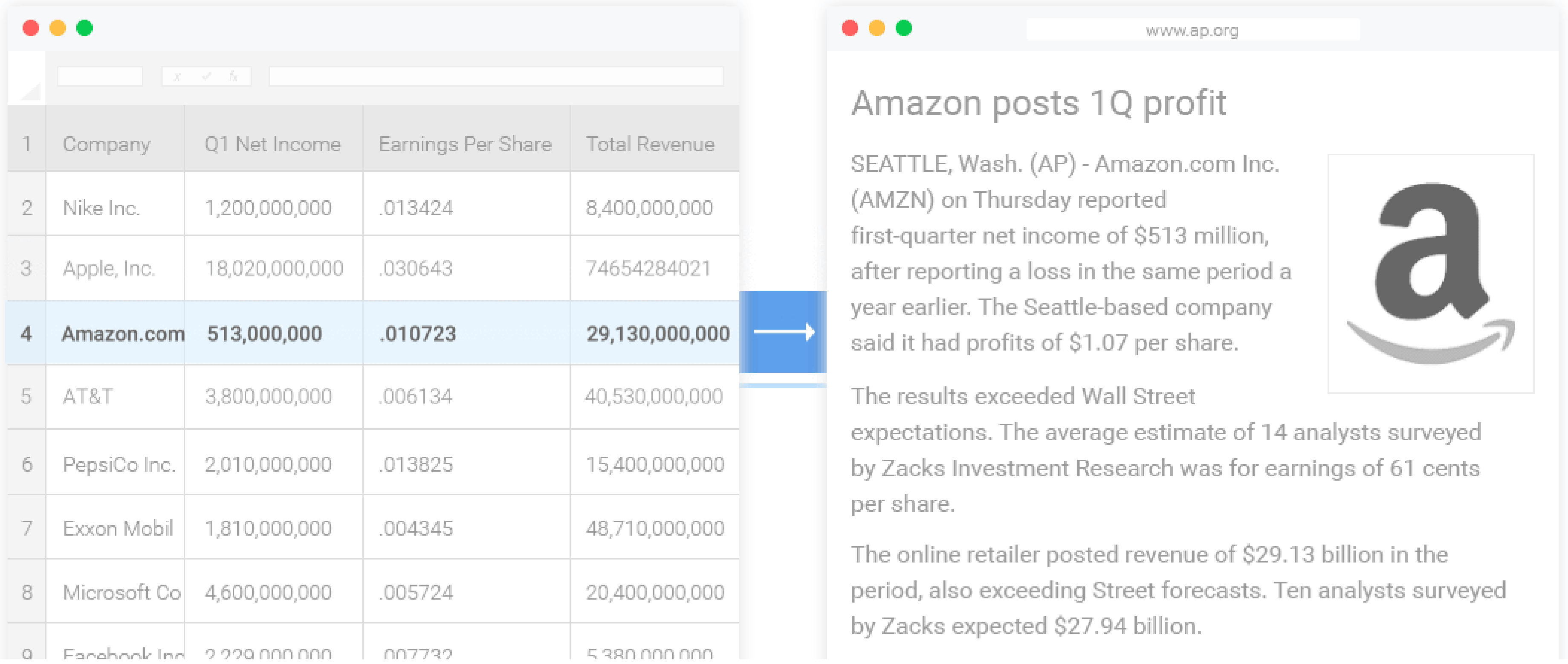
Well, I got it for two reasons. Number one, he won't want it. Probably a lobbyists, by the donors, and by the way, she was not awake to take that call at 3 o'clock in the morning.

And I want to tell you it's been no crowd like this.


Document Planning
content planning, document outline
Microplanning
referring expressions, word choice, aggregation
Realisation
converting specifications to a real text

Response Generation

- Usually **template** based with **rules** (e.g., code)



1	Company	Q1 Net Income	Earnings Per Share	Total Revenue
2	Nike Inc.	1,200,000,000	.013424	8,400,000,000
3	Apple, Inc.	18,020,000,000	.030643	74654284021
4	**Amazon.com**	**513,000,000**	**.010723**	**29,130,000,000**
5	AT&T	3,800,000,000	.006134	40,530,000,000
6	PepsiCo Inc.	2,010,000,000	.013825	15,400,000,000
7	Exxon Mobil	1,810,000,000	.004345	48,710,000,000
8	Microsoft Co	4,600,000,000	.005724	20,400,000,000
9	Facebook Inc	2,229,000,000	.007732	5,380,000,000

www.ap.org

Amazon posts 1Q profit

SEATTLE, Wash. (AP) - Amazon.com Inc. (AMZN) on Thursday reported first-quarter net income of $513 million, after reporting a loss in the same period a year earlier. The Seattle-based company said it had profits of $1.07 per share.

The results exceeded Wall Street expectations. The average estimate of 14 analysts surveyed by Zacks Investment Research was for earnings of 61 cents per share.

The online retailer posted revenue of $29.13 billion in the period, also exceeding Street forecasts. Ten analysts surveyed by Zacks expected $27.94 billion.

Response Generation

- The Conversational AI platform you use entails a specific method for generating responses

- Clinc: Jinja based on slots filled and intents
  - The weather in {% slots[‘city’] %} is {% slots[‘temperature’] %}.

- DialogFlow: Responses with syntax
  - The weather in $city is $temperature.

- Rasa: Built into stories
  - The weather in {city} is {temperature}.

Response Generation

- When you design the experience, a few considerations:
  - Is the response being **read aloud**? Or read on a **screen**?
  - Is the response going to be **interpreted** by some front-end?
    - Images?
    - HTML?
  - Does the response make **logical** sense?
    - Does it account for slots received already?
    - Does the response flow from previous responses?
  - Is the response an appropriate **length**?
    - Can the user find the information or do the task faster than it takes to read the response?