

BLS, Integrations, DialogFlow, and Rasa



Lecture 13

Review: NLP Fundamentals

- **Tokenization** is the process of splitting input strings into sequences of semantic tokens
 - Tokens should be lexemes in some language's lexicon (e.g., words)
- **Stemming** is a process of normalizing tokens
 - Remove some suffices
 - Allows easier comparison between words in an utterances
 - Allows more easily establishing relationships between words
- **Lemmatization** is the process of more easily relating words
 - Unconjugate verbs, etc.
- Resources like WordNet are established sources of semantic relationships

Review: NLP Fundamentals

- A **Language Model** is a probabilistic approach to predicting sequences of words
 - Text generation (e.g., the most likely word to use next, given a sequence)
 - ASR classification (e.g., the most likely transcription of audio)
 - Intent classification (e.g., the most likely intent given a sequence)
- We use **Part-of-speech tagging** to map individual tokens to underlying semantic information (e.g., verbs, nouns)
 - **Slot labeling** is a POS tagging problem: identify which tokens are slots
 - *Then*, classify which slot that token belongs to
- Statistics like tf-idf are used to establish relative importance of words
 - *Documents* can be thought of as *utterances*
 - *Terms* can be thought of as *tokens*

Review: Crowdsourcing

- **Crowdsourcing** is an approach to **rapidly gathering data**
 - Mechanical Turk (mTurk) is a common platform for crowdsourcing
 - You give a **prompt** to a user; workers complete them for money
 - (Clicn integrates mTurk, but you can use it separately)
- **Crowdworkers** are highly variable
 - They're paid chump change (\$0.10 - \$0.20 per task)
 - Incentivized to complete as many as possible
- Crowdworkers complete **Human Intelligence Tasks (HITs)**



Review: Crowdsourcing

- Crowdsourcing requires **trial-and-error**
 - **Brevity** in prompts is preferred
 - Don't gather all data at once—try multiple **prompts**
- Crowdworkers are easily biased
 - Prompt: "Rephrase 'What is the capitol of Florida?' without using X."

florida

what is the capital of FL

what is the capital of the sunshine state

what is the capital of the state where miami is located

what is the capital of the state that is located directly south of georgia

capital

hat city does the governor of florida live in

what is florida's statehouse city

where is the state government of florida headquartered

where is the seat of government in florida

what

i would like to know the capital of florida

can you tell me florida's capital

tell me the name of florida's capital

provide the name of the capital of florida

none

florida's capital is what

what city is the state capital of florida

what is florida's capital

can you name the capital of florida

Crowdsourcing

- Cline platform has crowdsourcing feature built-in
 - (and free to you)
- If you use DialogFlow or Rasa, you'll need to crowdsource yourself
 - Work with IAs to get mTurk up and running
 - We can also set up internal crowdsourcing (i.e., within the class)
- **Curation** still required
 - Don't just use crowdsourced data wholesale
 - Check for bots, cheaters, and low variability of responses

New Project

Select a customizable template to start a new project

Survey

Survey Link

Survey

Vision

Image Classification

Bounding Box

Semantic Segmentation

Instance Segmentation

Polygon

Keypoint

Image Contains

Video Classification

View instructions

Write what you would say in the given situation:

Context: You ordered a pair of shoes online and they don't fit

Intent: You want to return them

Type what you would say here...

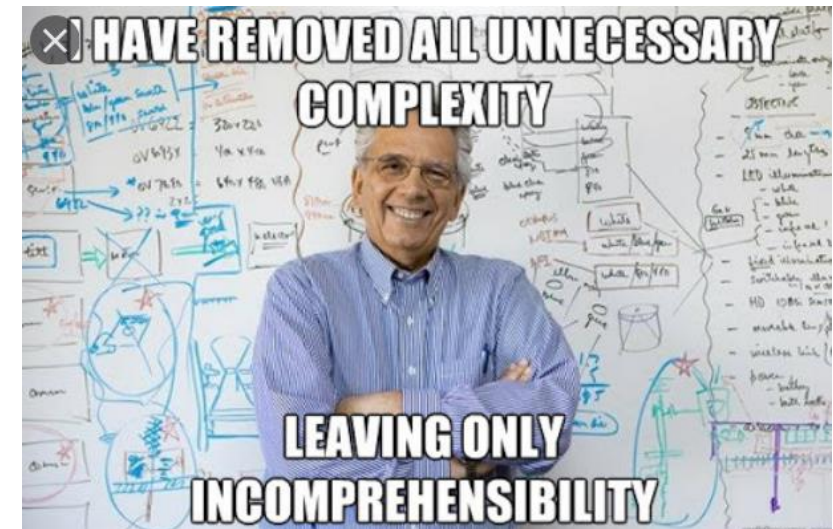
Submit

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
- Web services connect frontends, backends, and virtual assistants

Conversational AI Platforms

- Totally possible to have a simple command-line tool do this for us
 - We don't want to be constrained by a single system
 - How do others use a virtual assistant?
- Typically, platforms both:
 - **Expose an API** (engaged by your front-end)
 - **Engage an API** (engages your back-end)
- Details are platform-specific
 - But generally accomplish the same stuff



Web Services crash course

- So you've built a service
 - (e.g., a command-line virtual assistant)
 - How do you get others to be able to use it?
- Keep the service running, allow remote connections to engage it
 - Think: what happens when you go google.com?
- So: build some front-end, send a message to the computer running your service
 - Get a response, display it as appropriate

Web services crash course

- How is it done in practice?
 - Send JSON objects back and forth



```
{  
  "query" : "What's my account balance?",  
  "conversation_id": "Kevin",  
  other_metadata: { ... },  
}  
  
{  
  "response" : "U got $1 sry",  
  "conversation_id": "Kevin",  
  "intent": "account_balance"  
  ...  
}
```

Web Services Crash Course

- Conceptually identical to running a program with lots of arguments
 - `./chatbot.exe --query="What's my account balance?" --name="Kevin"`
- Remote service parses JSON payload, pushes it through a platform, and responds accordingly
 - Clinc, DialogFlow, Rasa all operate web services based on JSON
- Key takeaway:
 - Web service helps with scale and access to the service (e.g., a chatbot)

Comparing Platforms

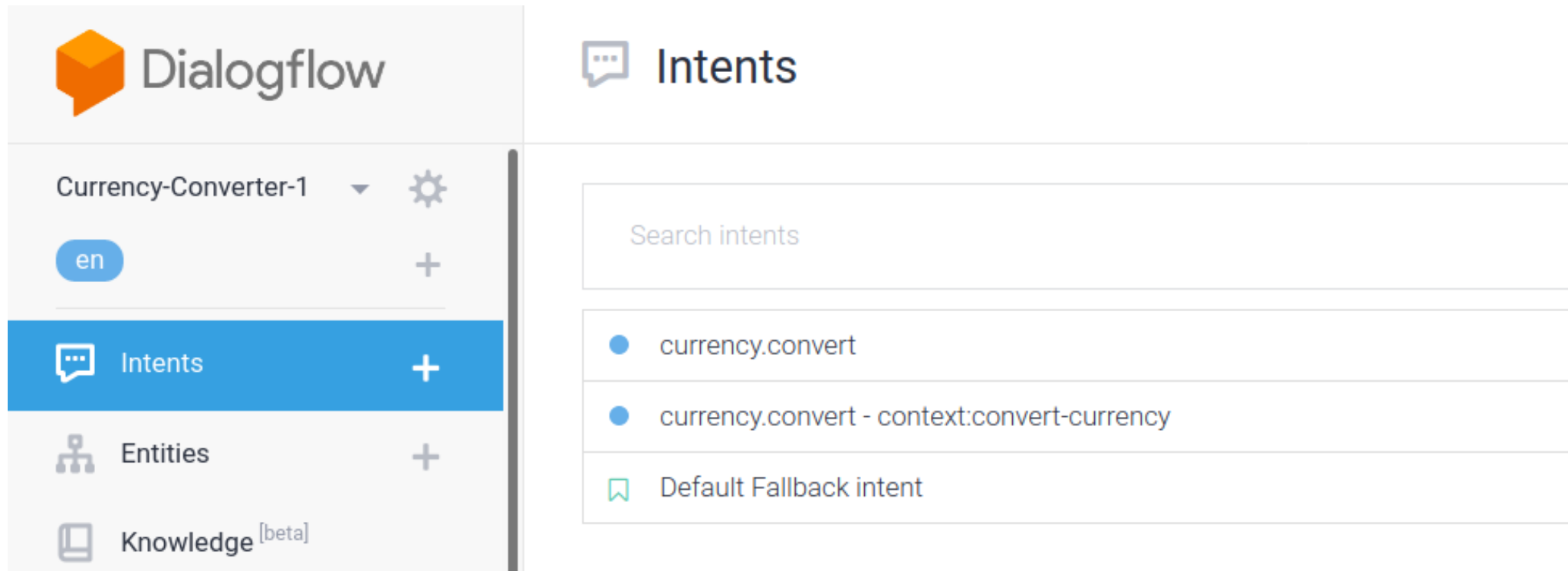
- Large software systems come with various tradeoffs
- For Conversational AI, platforms balance
 - Speed of development
 - Ease of experience design
 - Testing
 - Training time
 - End-user experience

Clinic

- 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

Google DialogFlow

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

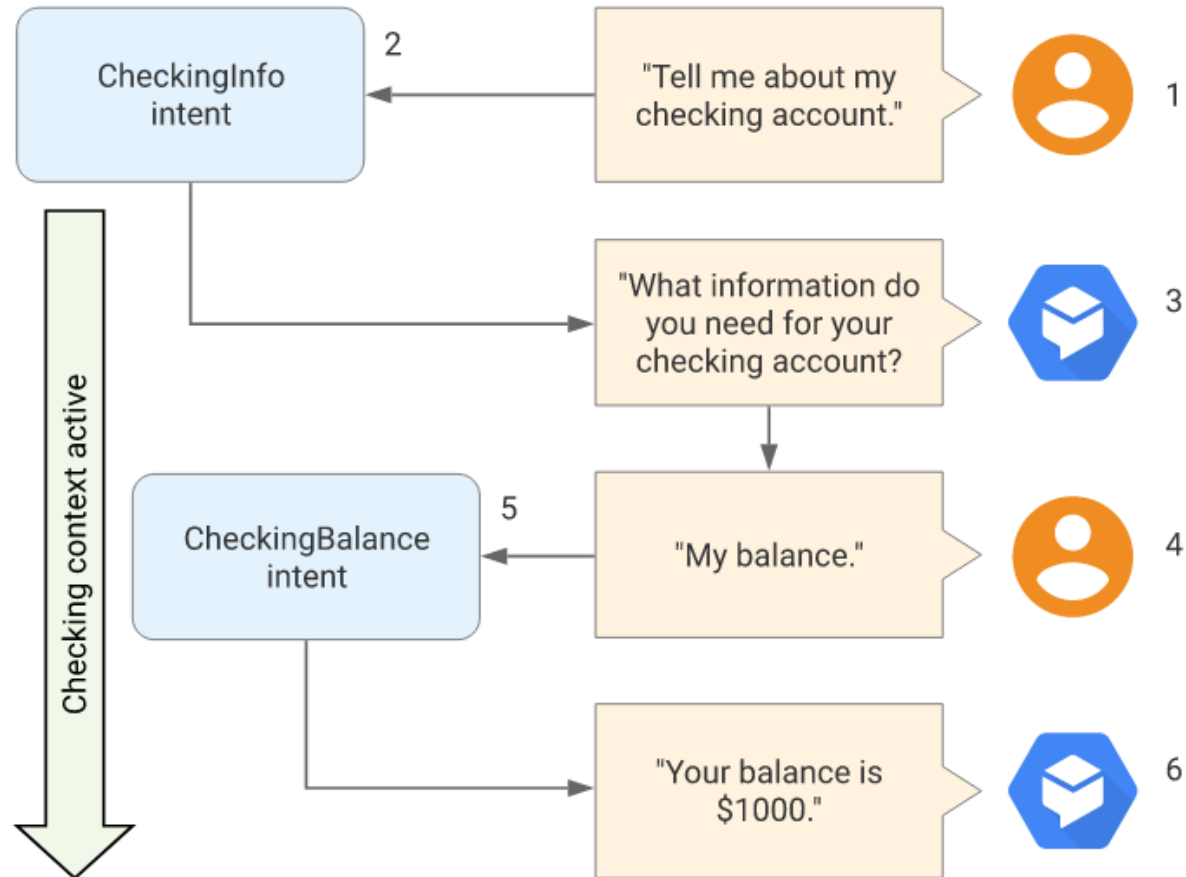


The screenshot displays the Google DialogFlow interface. On the left is a navigation sidebar with the DialogFlow logo at the top. Below the logo, the current project is identified as 'Currency-Converter-1' with a dropdown arrow and a settings gear icon. Underneath, the language is set to 'en' with a plus sign to add more languages. The 'Intents' menu item is highlighted in blue and includes a plus sign to add new intents. Other menu items include 'Entities' and 'Knowledge [beta]'. The main content area on the right is titled 'Intents' and features a search bar labeled 'Search intents'. Below the search bar, a list of intents is shown, each with a radio button for selection:

- 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 Cline **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

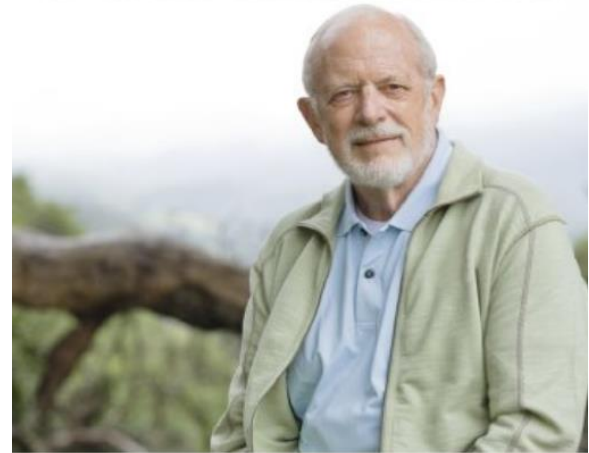
DialogFlow Contexts

- 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: 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

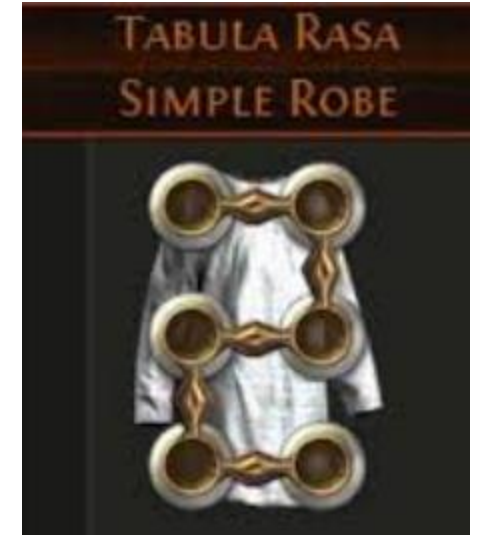
**IN PURSUIT OF
HAPPINESS AND
FULFILLMENT**



by Calvin A. Colarusso, M.D.

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

```
1 intents:
2   - greet
3   - goodbye
4   - affirm
5   - deny
6   - mood_great
7   - mood_unhappy
8   - bot_challenge
9
10 actions:
11 - utter_greet
12 - utter_cheer_up
13 - utter_did_that_help
14 - utter_happy
15 - utter_goodbye
16 - utter_iamabot
17 - action_myendpoint
18
19 responses:
20   utter_greet:
21     - text: "Hey! How are you?"
22
23   utter_cheer_up:
24     - text: "Here is something to cheer you up:"
25     image: "https://i.imgur.com/nGF1K8f.jpg"
26
```

Rasa: Intents

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

```
1 ## intent:greet
2 - hey
3 - hello
4 - hi
5 - good morning
6 - good evening
7 - hey there
8
9 ## intent:goodbye
10 - bye
11 - goodbye
12 - see you around
13 - see you later
14
15 ## intent:affirm
16 - yes
17 - indeed
18 - of course
19 - that sounds good
20 - 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

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)

