Desarrollo de Chatbots con Rasa

Rafa Haro Ramos
Antonio David Pérez Morales

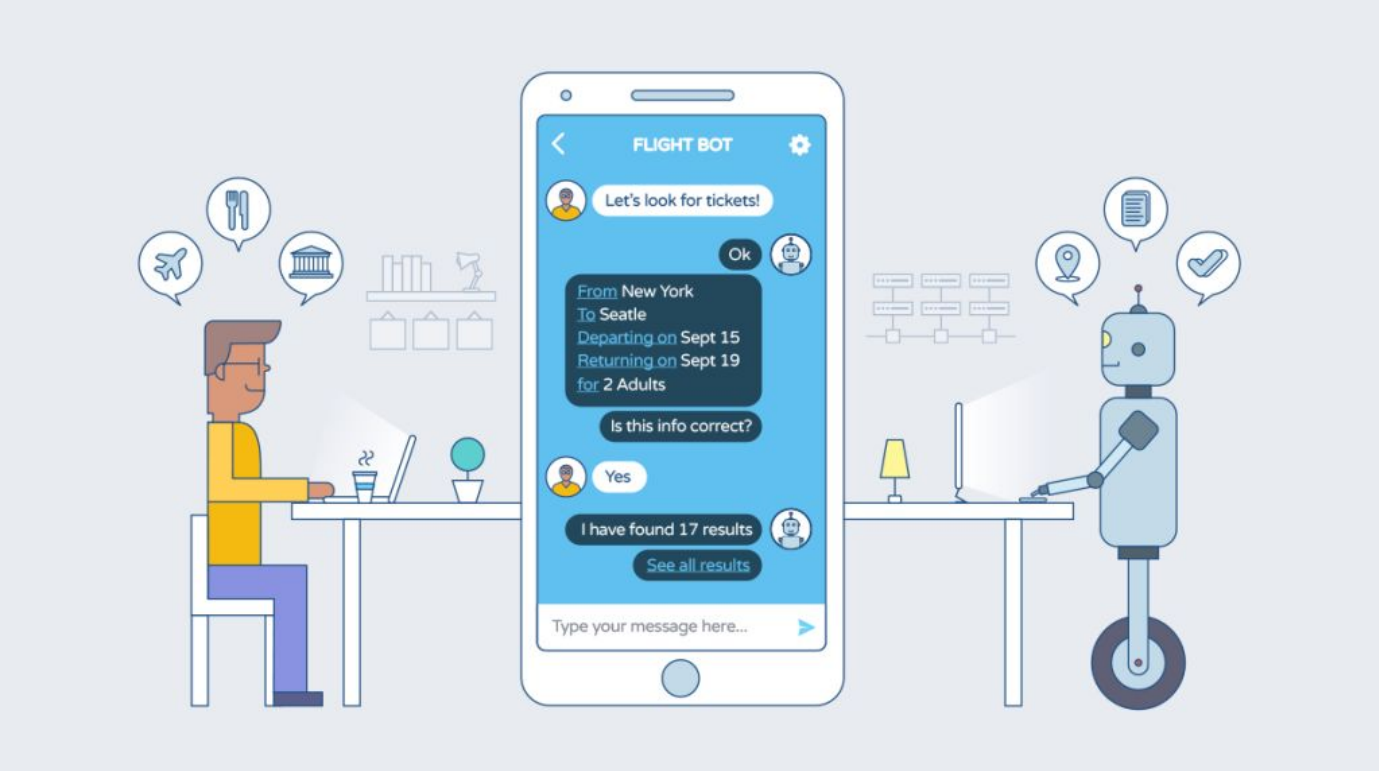
@rafa_haro
@adperezmorales

rafa_haro
adperezmorales

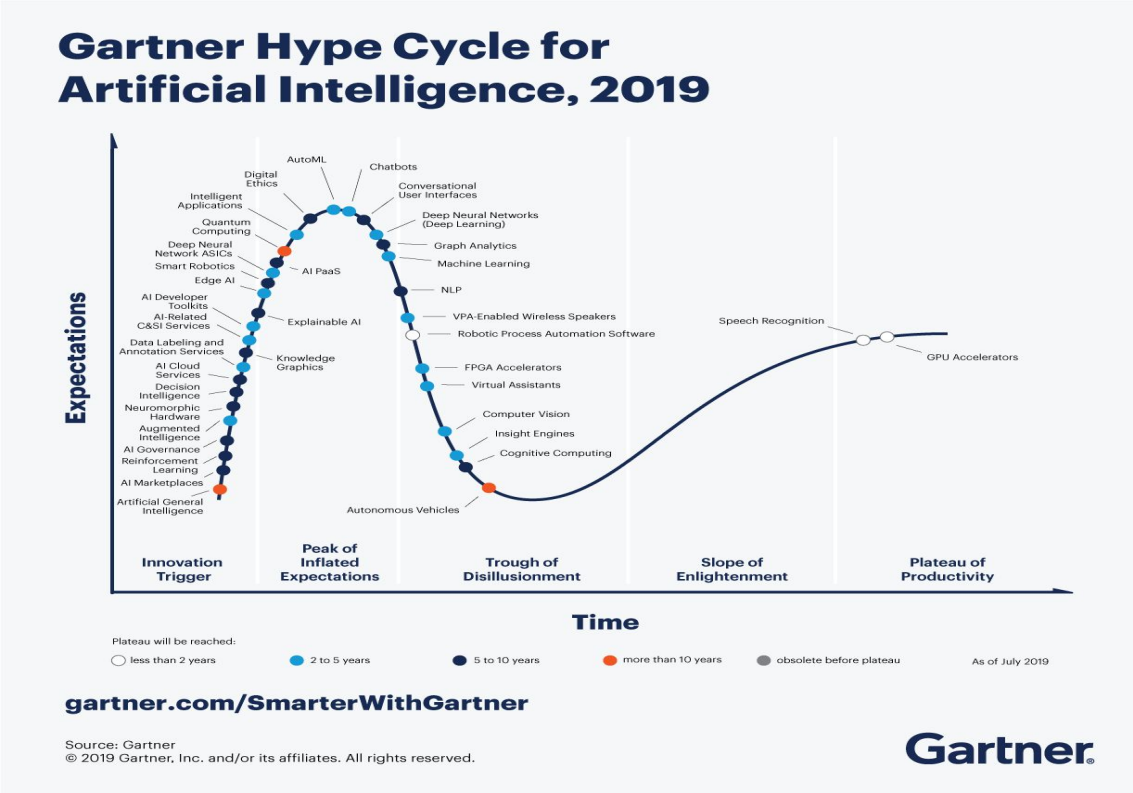
Agenda

- Chatbots Overview
- What is Rasa?
- Demo
- Rasa Architecture
- Starting a Project with Rasa
- Rasa NLU
- Rasa Core
- Interactive Learning
- Channels

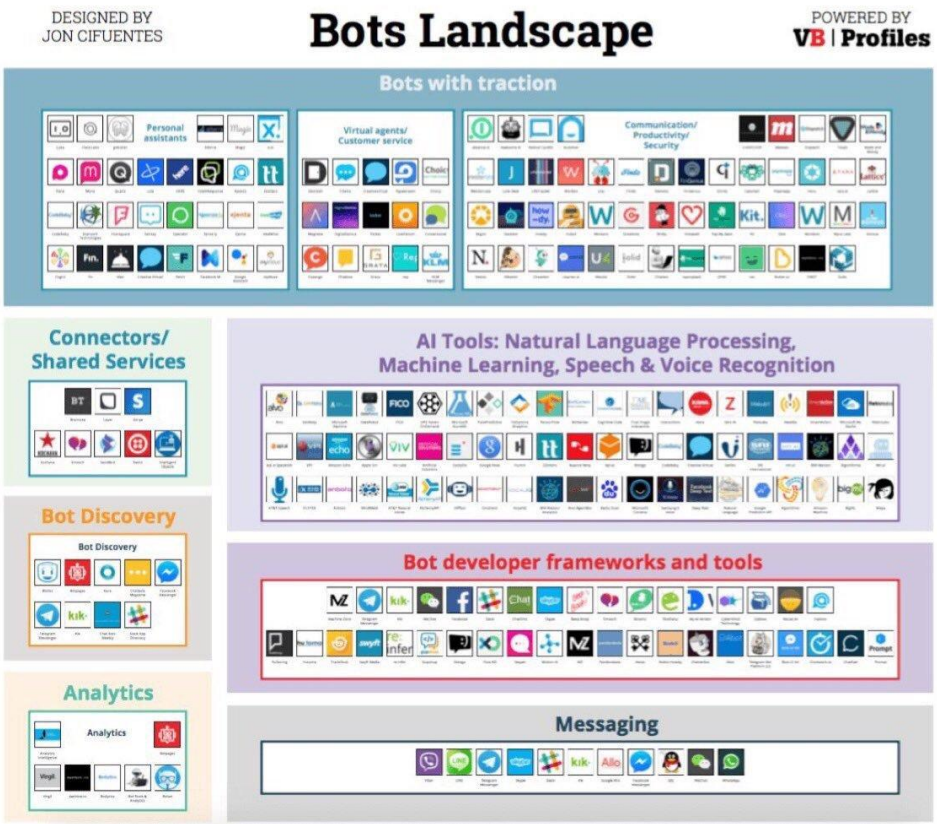
What's a Chatbot?



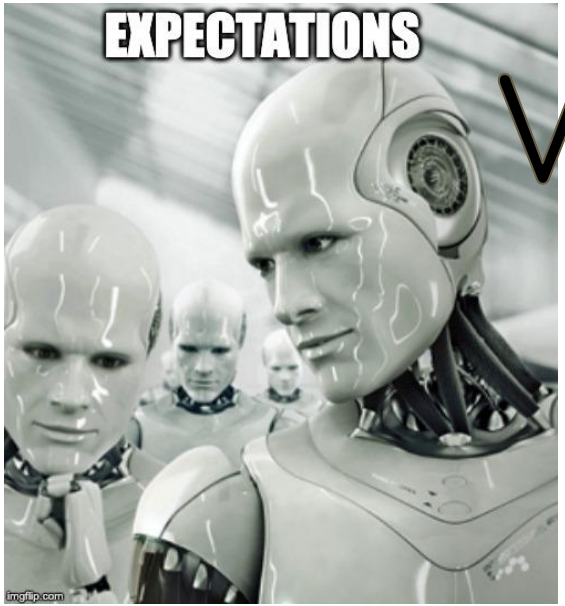
Hype, Hype Everywhere



Chatbots Landscape



Disclaimer: Manage The Expectations



Vs

Rasa Framework

- **Rasa** is an open source machine learning framework for automated text and voice-based conversations. Understand messages, hold conversations, and connect to messaging channels and APIs.

- Backed by a company of the same name

- Mature Community:
  - 500K+ Downloads
  - 3500K+ Forum Members
  - 300+ Contributors

Why Rasa?

- **Open Source**
  - Customizable
  - Community
  - Tune models for your use case
- **Own your data**
- **Avoid Vendor Lock-In**
- **Work locally**

Demo

PyconES 2019 Bot

bot

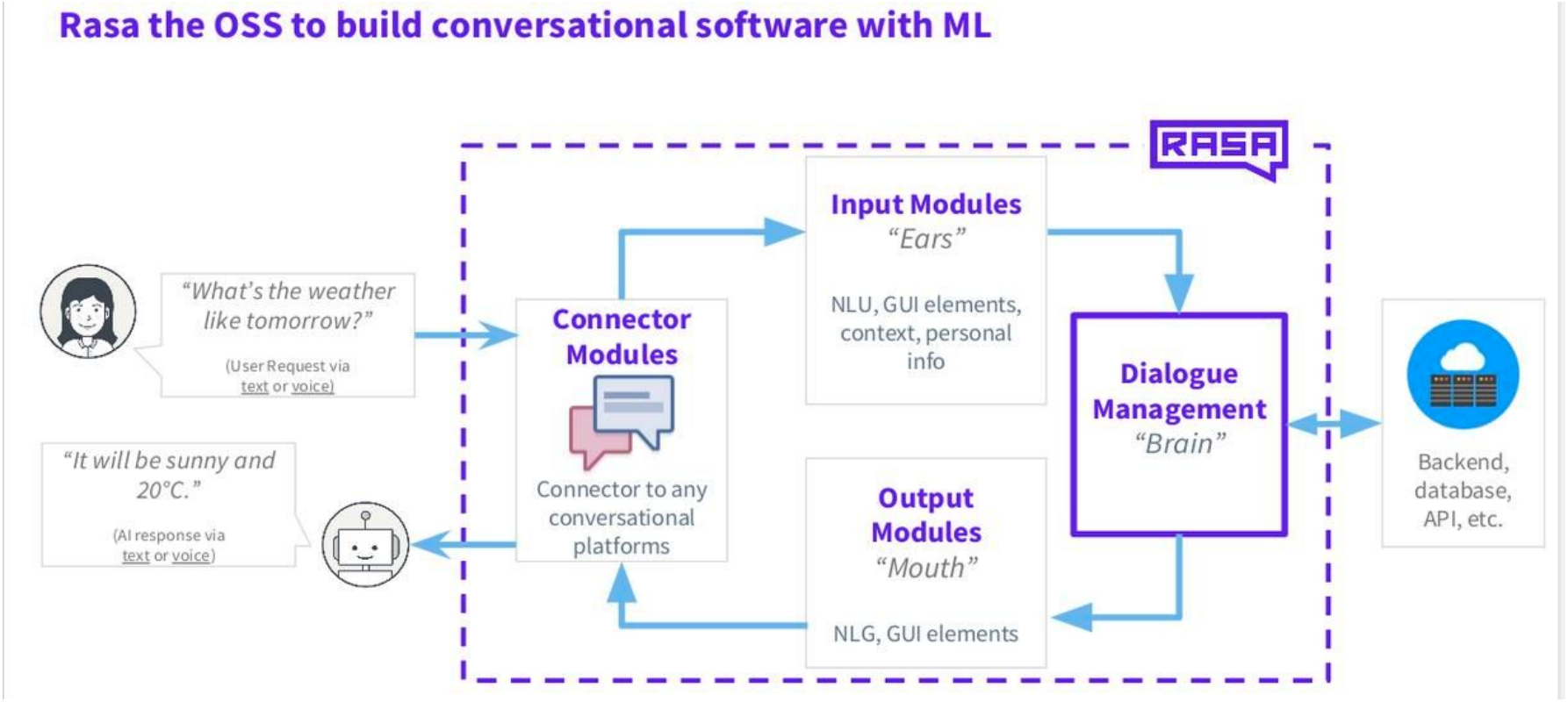
username

@Pycones19_bot

Puedes preguntarme acerca de las charlas de la pycon. Puedes hacerme preguntas como:

* Cuales son las siguientes charlas?
* A qué hora es la charla de Antonio?
* Hay alguna charla sobre microservicios?
* Charlas del sábado a las 17:30

Rasa Architecture



Starting a Project with Rasa

- rasa init

`__init__.py`	an empty file that helps python find your actions
`actions.py`	code for your custom actions
`config.yml` '*'	configuration of your NLU and Core models
`credentials.yml`	details for connecting to other services
`data/nlu.md` '*'	your NLU training data
`data/stories.md` '*'	your stories
`domain.yml` '*'	your assistant's domain
`endpoints.yml`	details for connecting to channels like fb messenger
`models/<timestamp>.tar.gz`	your initial model

Rasa NLU

- **NLU** stands for Natural Language Understanding
- Understand User Request (**Intent**)
- Converts user input text into a Data Structure, including extracted features
- Entities: most common features, but can be more
- NLU process can be rule based or ML based. Rasa is ML based

Rasa NLU: Select a Pipeline

- **Rasa provides two “macros”**

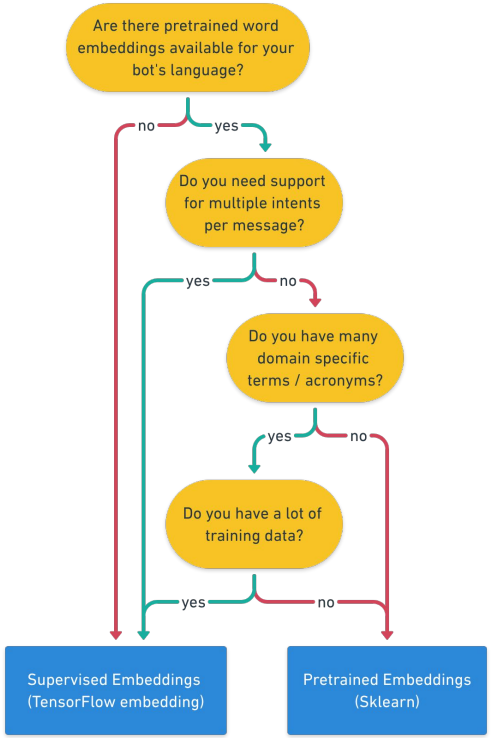
```
pipeline: "pretrained_embeddings_spacy"
```

```
pipeline:
- name: "SpacyNLP"
- name: "SpacyTokenizer"
- name: "SpacyFeaturizer"
- name: "RegexFeaturizer"
- name: "CRFEntityExtractor"
- name: "EntitySynonymMapper"
- name: "SklearnIntentClassifier"
```

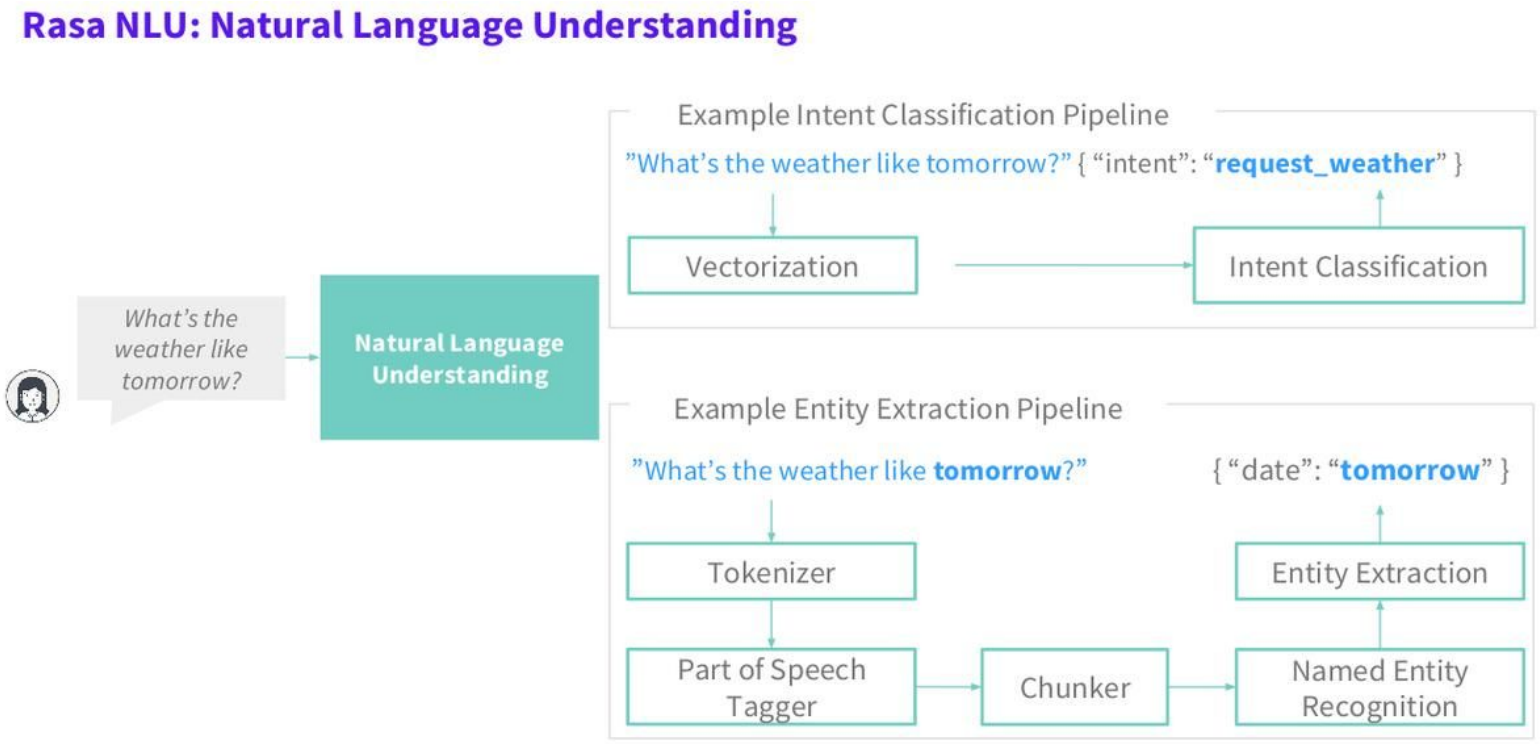
```
pipeline: "supervised_embeddings"
```

```
pipeline:
- name: "WhitespaceTokenizer"
- name: "RegexFeaturizer"
- name: "CRFEntityExtractor"
- name: "EntitySynonymMapper"
- name: "CountVectorsFeaturizer"
- name: "CountVectorsFeaturizer"
  analyzer: "char_wb"
  min_ngram: 1
  max_ngram: 4
- name: "EmbeddingIntentClassifier"
```

Rasa NLU: Select a Pipeline



Rasa NLU: Input Processing



Rasa NLU: Intents

```
## intent:saludo
- hola
- buenas
- hola, como estas?
- buenos días
- buenas tardes
- buenas noches
- que pasa
- saludos
```

```
## regex:time
- ([01]?[0-9]|2[0-3])(:[0-5][0-9])?

## lookup:day
- hoy
- mañana
- el sabado
- el domingo
- el sábado
- sábado
- domingo

## lookup:speaker
data/speakers.txt

## lookup:talk
data/talks.txt
```

```
## intent:a_que_hora
- a qué hora es la charla de [Rafa Haro](speaker)
- cuando es la charla de [Antonio David Perez](speaker)
- a qué hora presenta [María Marcos](speaker)
- cuándo es la charla de [Guillem Duran](speaker)
- cuando habla [Alberto](speaker)
- a que hora es la charla de [antonio](speaker)
- a qué hora es la presentación de [Orange](speaker)
- cuando presenta [Ravenpack](speaker)
- a que hora es la charla de [irene](speaker)
- cuando es la charla de [david garcia](speaker)
```

```
## intent:que_hay_a_las
- que charlas hay a las [11:30](time)?
- que hay a las [12](time)?
- que charlas hay [hoy](day) a las [17:00](time)
- que hay [mañana](day) a las [14:30](time)
- que charlas hay el [sabado](day) a las [11:00](time)
- charlas el [domingo](day) a la [13:00](time)?
- quién presenta el [sabado](day) a las [17](time)?
- que charlas hay el [sabado](day)
- charlas del [domingo](day)
- [hoy](day)
- [sabado](day)
- el [domingo](day)
- [12:10](time)
- [17](time)
- [1](time)
```

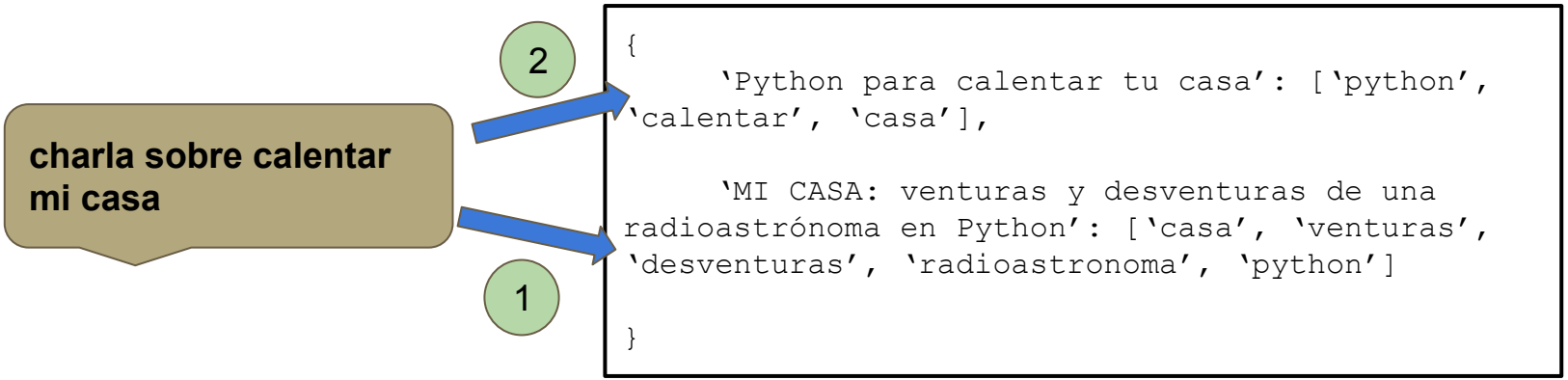
Rasa NLU: Entities

- Entity Extraction depends on the extractor configured in the Pipeline
- By default it trains a CRF NER
- Training data support **lookup tables**, regex and synonyms

Component	Requires	Model	Notes
`CRFEntityExtractor`	sklearn-crfsuite	conditional random field	good for training custom entities
`SpacyEntityExtractor`	spaCy	averaged perceptron	provides pre-trained entities
`DucklingHTTPExtractor`	running duckling	context-free grammar	provides pre-trained entities
`MitieEntityExtractor`	MITIE	structured SVM	good for training custom entities
`EntitySynonymMapper`	existing entities	N/A	maps known synonyms

Rasa NLU: Entities

- You can build your own Entity Extractor tailored to your use case
- For example: talks title extraction:
  - Dummy Frequency based extraction
  - Remove Stopwords
  - Normalize
  - Tokenize the input text
  - Select the candidates with the higher number of tokens in common



Rasa NLU Demo

Rasa Core: Stories

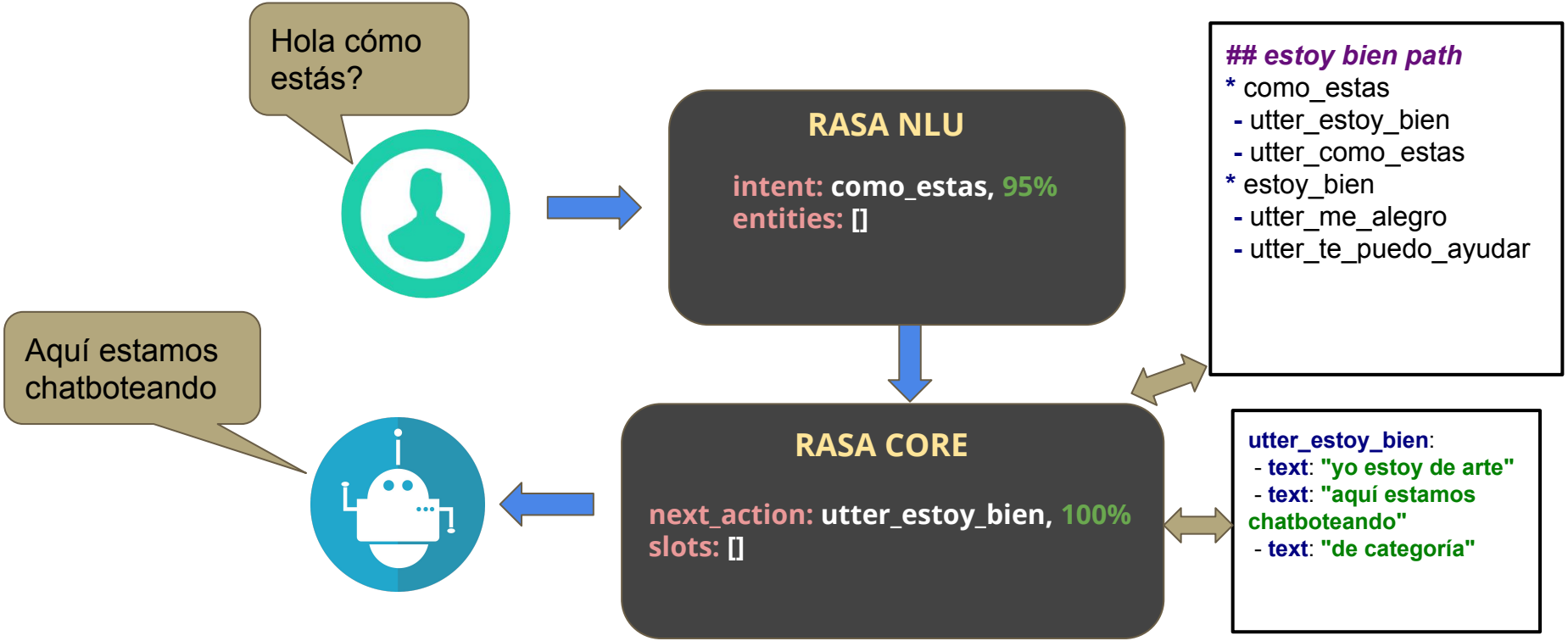
- **Conversations Training data for the chatbot**
- Rasa will decide the next action depending of this training data
- **Stories are made by a combination of Intents, Actions, Templates, Slots and Forms**
- Try to reflect as much conversations' paths as possible (Interactive Learning can help)
- You can feed your chatbot with conversation logs

Rasa Core: Stories

```
## say hello
* saludo
- action_hello

## say goodbye
* despedida
 - utter_despedida
 - action_restart

## estoy bien path
* como_estas
 - utter_estoy_bien
 - utter_como_estas
* estoy_bien
 - utter_me_alegro
 - utter_te_puedo_ayudar
```

```
templates:
 utter_saludo:
 - text: "Qué pasa pisha?"
 - text: "Qué pasa cabesa?"
 - text: "Qué pasa quillo?"

utter_despedida:
 - text: "venga nos vemos"

utter_estoy_bien:
 - text: "yo estoy de arte"
 - text: "aquí estamos chatboteando"
 - text: "de categoría"

utter_como_estas:
 - text: "¿y tu como andas?"
 - text: "¿y tu que te cuentas?"

utter_me_alegro:
 - text: "me alegro"
 - text: "wena esa"

utter_te_puedo_ayudar:
 - text: "¿qué te hace falta?"
 - text: "¿qué necesitas?"
 - text: "¿en qué te puedo ayudar?"
```

Rasa Core: Flow Diagram Example



Rasa Core: Slots

- **Key - Value store for your bot. It’s the bot memory**
- Can be used to store data gathered from the user (entities) or from the outside world (API, database...)
- Slots also influence next action decision
- Actions can fill slots are return Slot events

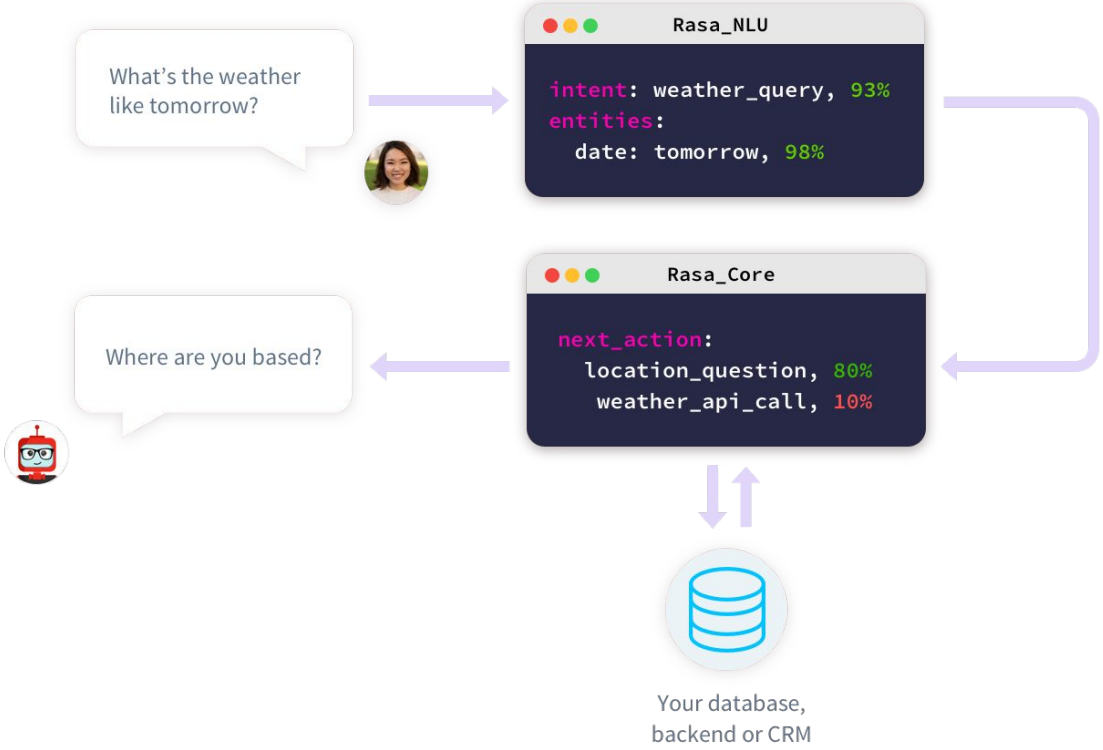
Rasa Core: Slots

```
## a que hora es la charla
* a_que_hora
 - slot{"speaker":"Rafa Haro"}
 - action_find_talk
 - action_listen
```

```
## que hay a las sin hora
* que_hay_a_las
 - action_find_talks_by_time{"day":"domingo"}
 - form{"name": "talk_form"}
 - action_find_talks_by_time{"time":"11:30","day":"domingo"}
```

Stories must reflect where the declared slots are filled

Rasa Core: Slots Example



Rasa Core: Actions

- **Actions are the things your bot runs in response to user input**

- Four kinds:
  - Utterance actions: start with utter_ and send a specific message to the user
  - Retrieval actions: start with respond_ and send a message selected by a retrieval model
  - Custom actions: run arbitrary code and send any number of messages (or none)
  - Default actions: e.g. action_listen, action_restart, action_default_fallback

- Custom Actions: backend query when the required slots are filled (Example: look for a talk when the bot knows time and day)

Rasa Core: Actions

```
class ActionFindNextTalks(Action):

  def name(self) -> Text:
    return "action_find_next_talks"

  def run(self, dispatcher: CollectingDispatcher,
          tracker: Tracker,
          domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

    now = datetime.now()
    search = Talk.search()
    query = search.sort('start').query('range', start={'gt': now})
    if query.count() == 0:
      dispatcher.utter_message("Pues ahora mismo no encuentro ninguna")
    else:
      message = "Las próximas charlas que he encontrado son:\n"
      message += range_query_to_message(query)
      dispatcher.utter_message(message)

    return []
```

Rasa Core: Forms

- **Interactive Slot Filling Strategy**
- Single action which contains the logic to loop over the required slots and ask the user for information
- The requested slots must be defined and the bot will keep going asking for them until all are filled. Unhappy paths can still be managed
- The message to ask for the next slot can be customized using utter_ask_{slot_name} templates

Rasa Core: Forms

Rasa Core: Forms

- **The way to fill the slot from user message can be customized:**
  - By default, from entities with the same name
  - Directly from text (input text is the slot value)
  - From an intent (if an intent is detected, fill the slot with a particular predefined value). This is useful for affirmations and negations
  - From a particular entity with a particular name

- Forms are active until filling all slots or programmatically deactivated. BE CAREFUL!

- On one hand, it could give the impression of having “memory” but, on the other, it could enter into an annoying loop

Rasa Workflow

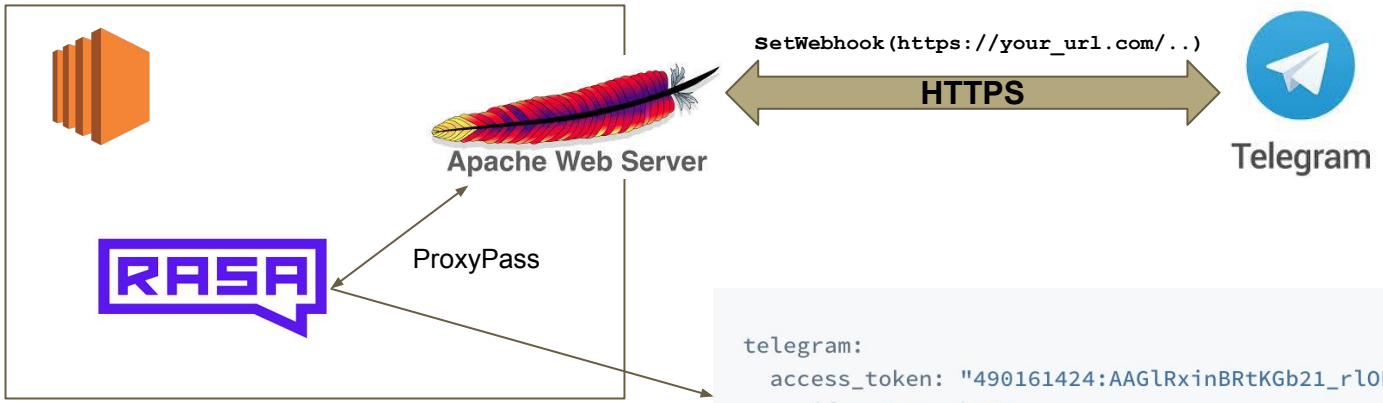
- rasa init
- rasa train
- rasa shell (--debug)
- rasa interactive

- rasa run actions
- rasa run

Rasa Interactive Demo

Rasa Channels

- **Rasa can ‘easily’ be used from several bot channels like Telegram, Slack, etc...**




```
telegram:
  access_token: "490161424:AAGlRxinBRtKGb21_rlOEMtDFZMXBl6EC0o"
  verify: "your_bot"
  webhook_url: "https://your_url.com/webhooks/telegram/webhook"
```

Rasa Channels: Telegram Requisites

- Communication must be over HTTPS
- You need valid signed certificates
- Amazon dynamic server names are not valid for signing a certificate. You must register a domain
- You must put apache in front of Rasa and communicate through a ProxyPass to Rasa HTTP Server

GRACIAS!

PYCON ES
ALICANTE 2019

@adperezmorales
@rafa_haro

adperezmorales
rafa_haro