Reconocimiento Automático del Habla

Práctica de chatbots con RASA

Curso 2023-2024

Qué es Rasa

- RASA es una plataforma de *código abierto* para construir chatbots avanzados.
- RASA Pro (comercial)
- Utiliza aprendizaje automático para comprender el lenguaje natural y llevar a cabo conversaciones más complejas.
- Ofrece herramientas para entrenar, construir y desplegar modelos de procesamiento de lenguaje natural y sistemas de diálogo, permitiendo a los desarrolladores crear sistemas conversacionales personalizadas y escalables.

Instalar Rasa (anaconda)

```
conda create --name rasa

conda activate rasa

conda install python=3.9
pip install rasa

# para utilizar el pipeline de spacy
pip install spacy

# para utilizar spacy en castellano
python -m spacy download es_core_news_md
```

Comandos Rasa

```
# entrenar el bot
rasa train

# iniciar del bot en modo consola, verbose
rasa shell -v

# iniciar el componente de nlu
rasa shell nlu

# iniciar el bot en modo servidor
rasa run actions

#muestra el flujo de diálogo
rasa visualize
```

Conceptos Básicos en Rasa

- intent: la intención del usuario (verbo)
- entity: la información que aporta (nombres)
- action: operaciones del bot
- responses: respuestas del bot
- stories: interacciones user/bot (dialog manager)
- rules: reglas de comportamiento (mini stories)
- domain: todo el universo del bot
- slots: la memoria del bot

Intent, entity y utterance tienen el mismo significado en Rasa que en Dialogflow. Los Fulfillments de Dialogflow se llaman Custom Action en Rasa.

Primer proyecto

crear un chatbot de muestra

```
rasa telemetry disable

rasa init --init-dir nombre_directorio

# o rasa init
```

ejecutar el servidor

```
rasa run
```

ejecutar modo consola

```
rasa shell
```

ver el contenido del directorio

```
tree nombre_directorio
```

Veamos brevemente alguno de los ficheros del chatbot.

config.yml

Define los componentes del chatbot y las *políticas* que aplicará el modelo en función de la entrada del usuario.

- recipe, puedes ser “default.v1” o “graph.v1”
- language, idioma del chatbot
- pipeline, *tubería* de componente para la comprensión del lenguaje natural (NLU). Entre los elementos que se pueden seleccionar está el core del modelo de lenguaje, el tokenizador, la representación densa de los turnos del usuario o el extractor de entidades. No es necesario definir una *pipeline* para tu chatbot, rasa elegirá automáticamente un *pipeline* al entrenar el modelo. `rasa train nlu` o `rasa train core`.

más información

`https://rasa.com/docs/rasa/model-configuration/`
`https://rasa.com/docs/rasa/nlu/components/`

config.yml

- policies, define la *estrategia* del chatbot, qué hacer (acción) en cada momento de la conversación con el usuario. Se puede optar por una estrategia basada en reglas y/o una estrategia basada en aprendizaje automático.

Se pueden definir varias estrategias para un mismo chatbot.

más información

`https://rasa.com/docs/rasa/policies`

Estrategia basada en reglas

`https://rasa.com/docs/rasa/policies/#rule-based-policies`

`https://rasa.com/docs/rasa/rules/`

Estrategia por aprendizaje automático

`https://rasa.com/docs/rasa/policies/#machine-learning-policies`

credentials.yml

Credenciales para plataformas de voz y texto

```
rest:
#  # you don't need to provide anything here - this channel doesn't
#  # require any credentials

#facebook:
#  verify: "<verify>"
#  secret: "<your secret>"
#  page-access-token: "<your page access token>"
...
# telegram:
#  access_token: "<access token>"
#  verify: "bot_name"
...
```

más información

https://rasa.com/docs/rasa/messaging-and-voice-channels

domain.yml

Define el dominio del chatbot. Hace un listado de todos los elementos que forman el sistema, incluyendo:

- intents. Lista de todos los intents que reconoce el chatbot.
- entities. Lista de todas las entities que se pueden extraer de los turnos de usuario (entity extractor).
- slots. Define la memoria del chatbot, pueden ser: texto, booleanos, reales, lista, categorías, o definidos por el usuario.
- responses, respuestas al usuario que envián un mensaje sin ejecutar código. `https://rasa.com/docs/rasa/responses`
- forms, formularios para recolectar información del usuario. `https://rasa.com/docs/rasa/forms`

domain.yml

- actions, cosas que puede hacer el chatbot como responder al usuario, consultar una base de datos o hacer una llamada a una API.
  `https://rasa.com/docs/rasa/actions`
- session_config. Configuración de la sesión. Una sesión es un diálogo entre un usuario y un chatbot.
- config, permite configurar el *store_entities_as_slots*.

más información

`https://rasa.com/docs/rasa/domain`

endpoints.yml

Definición de endpoints para el chatbot

```
# This file contains the different endpoints your bot can use.

# Server where the models are pulled from.
# https://rasa.com/docs/rasa/model-storage#fetching-models-from-a-
    server

#models:
#  url: http://my-server.com/models/default_core@latest
#  wait_time_between_pulls:  10   # [optional](default: 100)

# Server which runs your custom actions.
# https://rasa.com/docs/rasa/custom-actions

#action_endpoint:
#  url: "http://localhost:5055/webhook"

# Tracker store which is used to store the conversations.
# By default the conversations are stored in memory.
# https://rasa.com/docs/rasa/tracker-stores
```

data/nlu.yml

Ejemplos para los intents

```
version: "3.1"

nlu:
- intent: greet
  examples: |
    - hey
    - hello
    - hi
    - hello there
    - good morning
    - good evening
    - goodmorning
...
- intent: goodbye
  examples: |
    - cu
    - good by
    - cee you later
    - good night
    - bye
```

data/rules.yml

Definición de las reglas del chatbot

```
version: "3.1"

rules:

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
  steps:
  - intent: bot_challenge
  - action: utter_iamabot
```

data/stories.yml

Definición de las historias del chatbot. A diferencia de las reglas, las historias se pueden utilizar para aprender un modelo que generalice en conversaciones no vistas.

```
version: "3.1"
stories:

- story: happy path
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_great
  - action: utter_happy

- story: sad path 1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_unhappy
  - action: utter_cheer_up
  - action: utter_did_that_help
```

data/stories.yml

```
- story: sad path 2
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_unhappy
  - action: utter_cheer_up
  - action: utter_did_that_help
  - intent: deny
  - action: utter_goodbye
```

más información

https://rasa.com/docs/rasa/stories

actions/actions.py

Fichero python con código ligado a las acciones del chatbot

```
# This is a simple example for a custom action which utters "Hello
    World!"

# from typing import Any, Text, Dict, List
#
# from rasa_sdk import Action, Tracker
# from rasa_sdk.executor import CollectingDispatcher
#
#
```

actions/actions.py

```
# class ActionHelloWorld(Action):
#
#     def name(self) -> Text:
#         return "action_hello_world"
#
#     def run(self, dispatcher: CollectingDispatcher,
#             tracker: Tracker,
#             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
#
#         dispatcher.utter_message(text="Hello World!")
#
#         return []
```

más información

https://rasa.com/docs/rasa/custom-actions