

Explicación del desarrollo en curso del chatbot de Francisco José Checa Medina para la FCT

Se trata de un chatbot aún en desarrollo que simula un asistente para el caso ficticio de una página web que está creada por un local o empresa enfocada en la venta y consumo de sus productos, en este caso cervezas. El resultado final será la mejora y actualización del chatbot para que pueda dar respuestas más precisas y que logre simular el rol para lo que está pensado, junto con su posterior implementación en una página web que se subirá a AWS.

Desarrollo del Chatbot:

Realizado con Python y empleando RASA para brindar un conjunto de herramientas de código abierto para el desarrollo de chatbots y asistentes virtuales. Proporcionando así una especie de librería que sirve de plataforma para construirlos y entrenarlos, la idea me vino durante el desarrollo de un proyecto de PHP que hicimos antes de las navidades, vi que en un entorno de producción podía ser caótico si un usuario quería ver ciertas características de un producto, ya que en un caso práctico podría haber una base de datos bastante grande y el usuario tendría que estar bajando y abriendo pestañas buscando algún producto con unas características concretas, para este problema supuse que un chatbot podría ayudar al usuario a realizar sus búsquedas aún más rápido y ahorrarle tiempo. De este modo el chatbot realiza consultas a una base de datos que tiene implementada, en estos momentos se trata de una base de datos aún bastante simple porque primero quería entrenarlo en su día antes de seguir subiendo un escalón más, cuando el usuario pregunta sobre el país de una cerveza, su graduación alcohólica, entre otras opciones, el chatbot le devuelve respuestas coherentes, sin embargo, lamentablemente los tres archivos de python de "actions", sql_aux, y las validaciones dieron algunos errores, aún están pendientes de ser corregidas, como solución temporal comenté el código para que al menos el chatbot pudiese realizar algunas respuestas al usuario.

ACTIONS:

Para acciones y validaciones importé librerías de RASA, el planteamiento que tuve fue que el chatbot realizase las consultas en el "sql_aux".py. Por otra parte, "actions.py" sirve para crear acciones personalizadas en el chatbot en respuesta a ciertas intenciones detectadas por el usuario. El código además define varias clases, cada una de las cuales representa una acción personalizada. Cada clase tiene dos métodos principales: name y run.

Por último "validaciones.py" lo hice para crear validaciones personalizadas en el chatbot. Las validaciones personalizadas permiten verificar la información proporcionada por el usuario antes de que se procese, también se encuentran en proceso de mejora.

DATA:

- En data encontramos tres archivos con extensión .yaml (YAML Ain't Markup Language"). Es un formato de serialización de datos legible por humanos, estos archivos son:
- **nlu.yaml:** Siendo "nlu" las siglas de "Natural Language Understanding" es aquí donde se recogen las preguntas y acciones del usuario, y se guardan en un intent que cumple la función de una "intención" que tiene el usuario, tenemos algunas básicas de saludos y despedidas, y algunas que están relacionadas con la base de datos.

```
data > Y nlu.yaml
1  version: "3.1"
2  nlu:
3    - intent: saludo
4      examples: |
5        - hey
6        - hola
7        - hola, ¿qué tal?
8        - hey, hola
9        - buenos días
10       - buenas tardes
11       - buenas noches
12       - buenas
13       - Hola!
14    - intent: despedida
15      examples: |
16        - adiós
17        - hasta luego
18        - pasa un buen día
19        - nos vemos luego
20        - nos vemos más tarde
21        - hablamos luego
22        - nos vemos
23        - vale, adiós
24        - de acuerdo, hasta luego
25        - gracias, adiós
26        - de acuerdo, adiós
```

```
- intent: categorias
examples: |
- De que categoria es la cerveza [ipa](categoria)
- ¿Qué tipo de cerveza es [lager](categoria)?
- Me gustaria saber de que tipo es la cerveza [ale](categoria)
- ¿Puedes decirme que es la cerveza [scotch ale](categoria)?
- Quiero saber de que tipo es la [brown ale](categoria)?
- Dime de que es la cerveza [amber ale](categoria)?

- intent: grado_alcohol
examples: |
- Dime el grado de alcohol de la cerveza [ipa](tipo)
- Quiero saber la graduación que tiene la [lager](tipo)
- Me gustaria saber la graduación que tiene la cerveza [ale](tipo)
- Cuanta graduación tiene la cerveza [stout](tipo)
- Necesito saber cuantos grados tiene la cerveza [pilsner](tipo)
- Quisiera saber la graduación de la [pale ale](tipo)
- ¿Puedes decirme cuantos grados tiene la cerveza [weissbier](tipo)?
- Quiero qe me digas la graduación que tiene la cerveza [trappist](tipo)

- intent: precios
examples: |
- quiero ver los precios de las cervezas
```

- **rules.yaml:** Aquí vienen definidas las reglas para especificar el comportamiento del chatbot en respuesta a ciertos mensajes del usuario.

```

rules.yml
version: "3.1"

rules:

- rule: Saludar cuando el usuario salude.
  steps:
    - intent: saludo
    - action: utter_saludo
    - action: utter_ask_edad

- rule: Decir adiós cuando el usuario diga adiós.
  steps:
    - intent: despedida
    - action: utter_despedida

- rule: Decir que no entiende la frase
  steps:
    - intent: nlu_fallback
    - action: utter_no_entiendo

- rule: Decir que no entiende la frase cuando sea out of scope
  steps:
    - intent: out_of_scope
    - action: utter_no_entiendo

```

- **stories.yml:** A grandes rasgos son distintos escenarios que usará el chatbot en ciertas situaciones para usar como guía cuando interactúe con el usuario.

```

- story: Consulta cervezas
  steps:
    - intent: saludo
    - action: utter_saludo
    - action: utter_ask_edad
    - intent: respuesta_edad
    entities:
      - edad: '56'
    - slot_was_set:
      - edad:
        - '56'
    - action: action_edad
    - intent: Cervezas
    - action: action_cervezas

```

Fuera de DATA:

config.yml: Podremos configurar este apartado para cambiar algunos parámetros

```
1 # Configuration for Rasa NLU.  
2 # https://rasa.com/docs/rasa/nlu/components/  
3 language: "es"  
4  
5 pipeline:  
6   - name: "SpacyNLP"  
7     model: "es_core_news_sm"  
8     - name: "SpacyEntityExtractor"  
9       dimensions: ["LOC"]  
10  
11   - name: "WhitespaceTokenizer"  
12     # RegexFeaturizer y RegexEntityExtractor son necesarios para las regular expressions (email)  
13     - name: "RegexFeaturizer"  
14     - name: "RegexEntityExtractor"  
15     use_lookup_tables: false  
16     - name: "SpacyTokenizer"  
17     # CRFEntityExtractor es necesario para los lookups  
18     - name: "CRFEntityExtractor"  
19     BILOU flag: false
```

credentials.yml: almacena las credenciales necesarias para que tu bot interactúe con diferentes plataformas de chat y voz.

```
15 #slack:  
16 # slack_token: "<your slack token>"  
17 # slack_channel: "<the slack channel>"  
18 # slack_signing_secret: "<your slack signing secret>"  
19  
20 #socketio:  
21 # user_message_evt: <event name for user message>  
22 # bot_message_evt: <event name for bot messages>  
23 # session_persistence: <true/false>  
24  
25 socketio:  
26   user_message_evt: user_uttered  
27   bot_message_evt: bot_uttered  
28   session_persistence: false  
29  
30 #mattermost:  
31 # url: "https://<mattermost instance>/api/v4"  
32 # token: "<bot token>"  
33 # webhook_url: "<callback URL>"  
34  
35 # This entry is needed if you are using Rasa X. The entry represents credentials  
36 # for the Rasa X "channel", i.e. Talk to your bot and Share with guest testers.  
37 rasa:  
38   url: "http://localhost:5002/api"  
39
```

domain.yml: Es un marco de trabajo para el desarrollo de chatbots. Define las opciones en las que opera el bot, incluyendo las intenciones que puede entender, las entidades que puede extraer, las respuestas que puede dar y las acciones que puede realizar.

```
Y domain.yml  
1 version: '3.1'  
2 session_config:  
3   session_expiration_time: 60  
4   carry_over_slots_to_new_session: true  
5 intents:  
6   - Cervezas  
7   - afirmacion  
8   - categorias  
9   - despedida  
10  - grado_alcohol  
11  - negacion  
12  - out_of_scope  
13  - paises  
14  - precios  
15  - respuesta_edad  
16  - saludo  
17 entities:  
18  - edad  
19  - tipo  
20  - pais  
21 slots:  
22  edad:  
23    type: list  
24    mappings:  
25    - type: from_entity  
26      entity: edad  
27  pais:  
28    type: list
```

endpoints.yml: Define los diferentes endpoints que el chatbot podrá usar

```

1 # This file contains the different endpoints your bot can use.
2
3 # Server where the models are pulled from.
4 # https://rasa.com/docs/rasa/model-storage#fetching-models-from-a-server
5
6 #models:
7 # url: http://my-server.com/models/default\_core@latest
8 # wait_time_between_pulls: 10 # [optional](default: 100)
9
10 # Server which runs your custom actions.
11 # https://rasa.com/docs/rasa/custom-actions
12
13 action_endpoint:
14   url: "http://localhost:5055/webhook"
15
16 # Tracker store which is used to store the conversations.
17 # By default the conversations are stored in memory.
18 # https://rasa.com/docs/rasa/tracker-stores
19
20 #tracker_store:
21 #   type: redis
22 #   url: <host of the redis instance, e.g. localhost>
23 #   port: <port of your redis instance, usually 6379>
24 #   db: <number of your database within redis, e.g. 0>
25 #   password: <password used for authentication>
26 #   use_ssl: <whether or not the communication is encrypted, default false>

```

Lanzar el chatbot:

Para lanzarlo nos iremos al terminal y pondremos el comando: `rasa run -m models --enable-api --cors "*"` . Este comando nos habilita una API HTTP que acepta solicitudes para interactuar con el modelo de Rasa, y permite el acceso desde cualquier origen a través de CORS.

```
C:\Users\Fran\Desktop> python models\20240213-211328-grizzled-radon.tar.gz
Your Rasa model is trained and saved at 'models\20240213-211328-grizzled-radon.tar.gz'.
PS C:\Users\Fran\Desktop\CerveBot-FCT> rasa run -m models --enable-api --cors "*" --
C:\Users\Fran\AppData\Local\Programs\Python\Python39\lib\site-packages\rasa\core\tracker_store.py:1044: Move
```

- Para que ejecute acciones tendremos que abrir un segundo terminal y usar el comando: `rasa run actions`

```
PS C:\Users\Fran\Desktop\CerveBot-FCT> rasa run actions
C:\Users\Fran\AppData\Local\Programs\Python\Python39\lib\site-packages\rasa\core\tracker_store.py:1044: MovedIn20Warning: Deprecated AP
I features detected! These feature(s) are not compatible with SQLAlchemy 2.0. To prevent incompatible upgrades prior to updating applic
```



- De esta forma si el chatbot detecta que somos menores de edad no nos dará permiso para continuar



- Sin embargo como vimos anteriormente la integración con la base de datos da fallos, pero algunos intents como el out_of_scope o el de despedida entre otros si funcionan



