
- TRABAJO FINAL -
Chatbot de recomendación de películas

Reconocimiento Automático del Habla

Autora

Aitana Menárguez Box

Enero 2024

Índice

1	Introducción	3
2	Funcionalidades del <i>chatbot</i>	3
2.1	Recomendación de películas	3
2.1.1	Código básico	3
2.2	Obtención de datos externos	4
2.2.1	API	4
2.2.2	CSV	4
3	Montaje del <i>chatbot</i>	5
3.1	Implementación en Telegram	5
4	Ejemplos de uso	5
5	Conclusiones y trabajos futuros	5

1 Introducción

El objetivo de este trabajo es investigar nuevas formas de usar la herramienta RASA para la creación de *chatbots*, partiendo de los conocimientos adquiridos en la segunda práctica de la asignatura. En concreto, se ha implementado un *chatbot* para la recomendación de películas y la extracción de la información relativa a éstas a partir de diferentes interacciones con el usuario. A lo largo de esta memoria se presenta la evolución del *chatbot* realizado, desde su fase inicial hasta las ampliaciones posteriores.

2 Funcionalidades del *chatbot*

Las funcionalidades del *chatbot* se han implementado a partir de las *actions*, *stories* y el *domain* de RASA. A continuación se explica cómo se ha construido la herramienta y qué opciones brinda al usuario.

2.1 Recomendación de películas

El objetivo es obtener los datos que envía el usuario para, a partir de ellos, encontrar una película o una serie de películas que cumplan con los requisitos especificados. Para ello, se han incorporado dos formas diferentes de pedir películas por parte del usuario:

- **Consulta guiada:** donde el usuario podrá elegir si desea que el *chatbot* le recomiende una película (de forma aleatoria) o si prefiere seleccionar 3 valores diferentes (género, año y nota) sobre la película para encontrar una que cumpla con los datos introducidos.
- **Consulta propia:** en este caso el usuario escribe de forma libre de qué director, año, nota o género quiere que el *chatbot* le recomiende la película.

2.1.1 Código básico

En el `domain.yaml` se han añadido las *entities* y *slots* de género, director, valoración y año sobre la película para que el *chatbot* pueda recuperar la información y realizar la búsqueda en cualquier momento. Además, dispondrá de una acción por cada tipo de búsqueda: búsqueda de género, director, valoración o año, búsqueda aleatoria o búsqueda según los valores introducidos por el usuario en la consulta guiada. Por otro lado, en las respuestas del *chatbot*, se han incorporado botones para facilitar la interacción de usuario y evitar que se cometan errores.

Aquí se puede ver un ejemplo de implementación de los botones:

```
1 utter_ask_year:
2   - text: "What year are you interested in?"
3     buttons:
4       - title: "1920-1950"
5         payload: '/year_movie>{"year": "1960"}'
6       - title: "1960-1980"
7         payload: '/year_movie>{"year": "1990"}'
8       - title: "1980-2000"
9         payload: '/year_movie>{"year": "2000"}'
10      - title: "2000 onwards"
11        payload: '/year_movie>{"year": "2010"}'
```

2.2 Obtención de datos externos

La obtención de los datos externos sobre películas para poder recuperar la información pedida por el usuario, se ha intentado implementar de dos formas distintas: por consultas a una API y por consultas a un fichero .csv local. Ambas opciones se han realizado sobre un *script python* de acciones (`actions.py`).

2.2.1 API

La API utilizada ha sido OMDb API (a través de <https://www.omdbapi.com/>), que permite consultas para recuperar información sobre películas dado su título o un identificador de IMDb. Es evidente que este tipo de recuperación de información no es siempre fiable, ya que limita el número de consultas que se pueden realizar al día y el mantenimiento de los datos depende de fuentes externas.

Un ejemplo del uso de la API se puede ver a continuación, donde se implementa la búsqueda de una película aleatoria. Se busca por un identificador de IMDb (2 letras y 7 números) generado aleatoriamente.

```
1 def random_imdb_movie():
2     # create random id with 9 digits, 2 letters and 7 numbers
3     id = ''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for i in range
4         (2)) + ''.join(random.choice('0123456789') for i in range(7))
5     params = {
6         'apikey': api_key,
7         'i': id
8     }
9     base_url = 'http://www.omdbapi.com/?'
10    url = base_url + urlencode(params)
11    response = requests.get(url)
12    data = response.json()
13    return data
```

Antes de poder realizar consultas, se ha tenido que registrar un correo electrónico en la web de la API para poder obtener una clave para poder usarla.

2.2.2 CSV

El fichero .csv se ha descargado desde <https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows?resource=download>. Esta opción, a pesar de tener que depender de un archivo estático, ha permitido implementar de forma más sencilla la búsqueda de películas según un criterio, ya que no depende de consultas y se pueden gestionar los datos de forma autónoma y en formato que se desee. Es por eso que se ha escogido esta opción para la implementación final del *chatbot*. Un ejemplo de cómo se ha recuperado la información puede verse en las siguientes líneas. Aquí, se recupera la información guardada para la consulta a través del `tracker.get_slot()`.

```
1 class ActionGiveGenre(Action):
2
3     . . .
4
5     def run(self, dispatcher: CollectingDispatcher,
6         tracker: Tracker,
```

```

7         domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
8
9         genre = tracker.get_slot("genre")
10        match_movies = []
11        for movie in MOVIESDB:
12            mv_genre = movie[5]
13            if genre in mv_genre.lower():
14                match_movies.append(movie)
15
16        if not match_movies: # if list is empty
17            msg = f"Sorry, I couldn't find any {genre} movies...\nYou can ask
18                  for another movie, tho :)"
19        else:
20            . . .

```

3 Montaje del *chatbot*

En primer lugar, es necesario permitir el tráfico con las acciones del código python (a través del fichero `endpoints.yml`). Con el comando `rasa run actions` se activará el script de acciones y se podrán ejecutar sin problema. Después de haber entrenado el modelo según los ejemplos incorporados y las historias añadidas (a partir de `rasa train`), se ha ejecutado el modelo.

3.1 Implementación en Telegram

Con el objetivo de que el *chatbot* sea más accesible a nivel de usuario, se ha decidido ejecutar a través de Telegram. En primer lugar, ha sido necesario crear un nuevo *chatbot* en Telegram a partir del `@BotFather`, un bot que facilita la gestión para la creación de *bots* en Telegram. Dentro de éste se pueden consultar las credenciales para poder ejecutar el *chatbot* de RASA a través de Telegram. Será en el archivo `credentials.yml` donde se incorporen estos datos, junto con la url obtenida a través de *ngrok* para permitir el tráfico de información a través de Telegram. Por último, es necesario activar el flag `--enable-api` a la hora de lanzar el *chatbot* de RASA.

4 Ejemplos de uso

En las figuras 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10 se muestran unas capturas con ejemplos de conversaciones con el *chatbot*.

5 Conclusiones y trabajos futuros

De cara al futuro, una de las posibles mejoras podría ser la aplicación de alguna base de datos que permita al *chatbot* reaccionar de forma más humana. Podría probarse a entrenar el modelo con información sobre textos con "sentimientos" para que así tuviera una percepción más precisa de las intenciones del usuario y recomendarle así películas de forma más adecuada. También de podría probar a ofrecer más información sobre la película

recuperada, como la disponibilidad según las plataformas de *streaming* disponibles en las cuales se pueda ver.

Tanto este trabajo como el anteriormente planteado, incorporan diferentes ámbitos de las tecnologías. Desde modelos de lenguaje, pasando por consultas a APIs o creaciones de bases de datos y programación en python. Hay muchas opciones por explorar y muchas de ellas pueden ser incorporadas en RASA.

```

Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> i want to watch a movie
? Do you have any special preferences about the movie? 2: No, you can decide (/deny{"guided": "False"})
Here you have your movie:
- Title: Rang De Basanti
- Director: Rakeysh Omprakash Mehra
- Year: 2006
- Rating: 8.2
- Overview: The story of six young Indians who assist an English woman to film a documentary on the freedom fighters from their past, and the events that lead them to relive the long-forgotten saga of freedom.
Image: https://m.media-amazon.com/images/M/MV5BYThmZDA0YmQzMWJhNy80MDQwLTk0Y2YtHDhmZTE5ZjhlNjllXkEjXkFqcGdeQWVyODU0EShzeF30Tg._V1_UY98_CR1,0,67,98_AL_.jpg
Your input -> 

```

Figura 1: Búsqueda de película guiada aleatoria

```

Your input -> i want to watch a movie
? Do you have any special preferences about the movie? 1: Yes, I do (/affirm{"guided": "True"})
? What genre are you interested in? (Use arrow keys)
1: Adventure (/genre_movie{"genre": "Adventure"})
2: Biography (/genre_movie{"genre": "Biography"})
3: Comedy (/genre_movie{"genre": "Comedy"})
4: Drama (/genre_movie{"genre": "Drama"})
» 5: Horror (/genre_movie{"genre": "Horror"})
6: Musical (/genre_movie{"genre": "Musical"})
7: Romance (/genre_movie{"genre": "Romance"})
8: Sci-Fi (/genre_movie{"genre": "Sci-Fi"})
9: Thriller (/genre_movie{"genre": "Thriller"})
10: Western (/genre_movie{"genre": "Western"})
Type out your own message...

```

Figura 2: Búsqueda de película guiada según criterios (I)

```

Your input -> i want to watch a movie
? Do you have any special preferences about the movie? 1: Yes, I do (/affirm{"guided": "True"})
? What genre are you interested in? 5: Horror (/genre_movie{"genre": "Horror"})
? What year are you interested in? (Use arrow keys)
1: 1920-1950 (/year_movie{"year": "1960"})
» 2: 1960-1980 (/year_movie{"year": "1990"})
3: 1980-2000 (/year_movie{"year": "2000"})
4: 2000 onwards (/year_movie{"year": "2010"})
Type out your own message...

```

Figura 3: Búsqueda de película guiada según criterios (II)

```
? What genre are you interested in? 5: Horror (/genre_movie{"genre": "Horror"})
? What year are you interested in? 2: 1960-1980 (/year_movie{"year": "1990"})
? What rating do you want the movie to have? 2: 8 (/rating_movie{"rating": "8"})
Here you have your movie:
- Title: The Shining
- Director: Stanley Kubrick
- Year: 1980
- Overview: A family heads to an isolated hotel for the winter where a sinister presence influences the father into violence, while his psychic son sees horrific forebodings from both past and future.
Image: https://m.media-amazon.com/images/M/MV5BZWFlYmY2MGZjVjVjYS00YzU4LTg0YjQtYzY1ZGE3NTA5NGQxXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_UX67_CR0,0,67,98_AL_.jpg
Your input -> 
```

Figura 4: Búsqueda de película guiada según criterios (III)

```
Your input -> i want to watch a Sofia Coppola movie
Here are some movies from Sofia Coppola:
- Lost in Translation BY Sofia Coppola
Your input -> 
```

Figura 5: Búsqueda de película libre por director

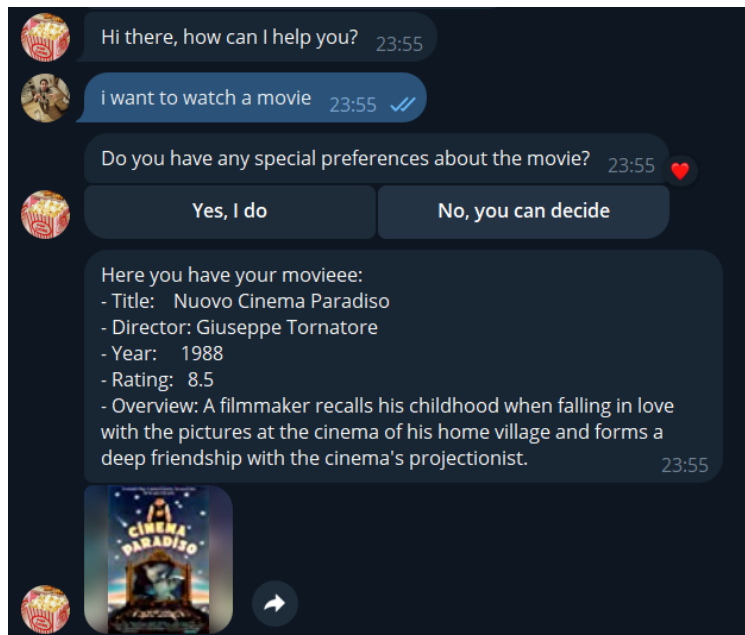


Figura 6: Búsqueda guiada aleatoria en Telegram

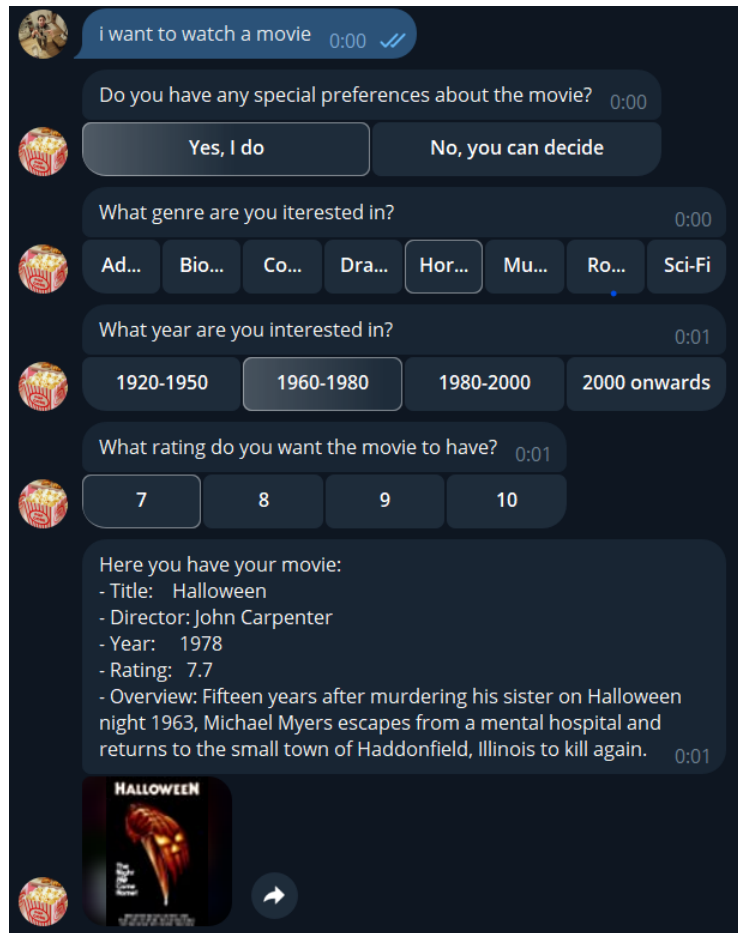


Figura 7: Búsqueda de película guiada según criterios en Telegram

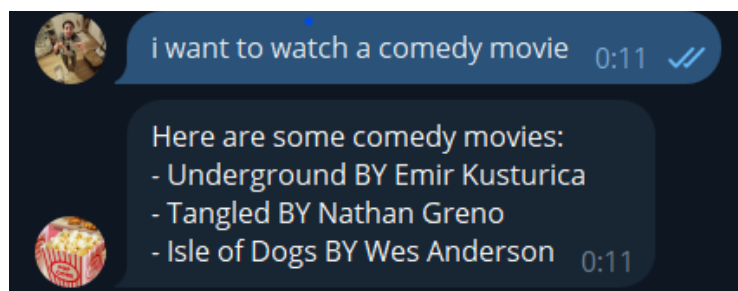


Figura 8: Búsqueda de película libre por género en Telegram

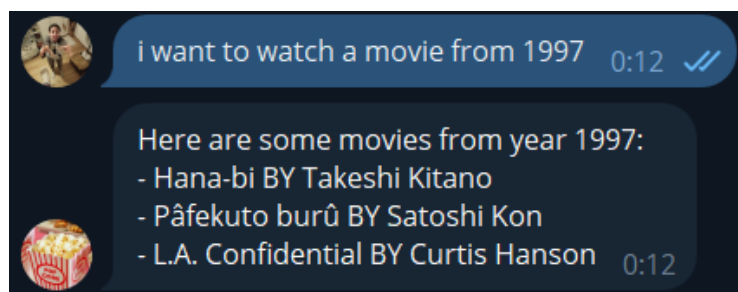


Figura 9: Búsqueda de película libre por año en Telegram

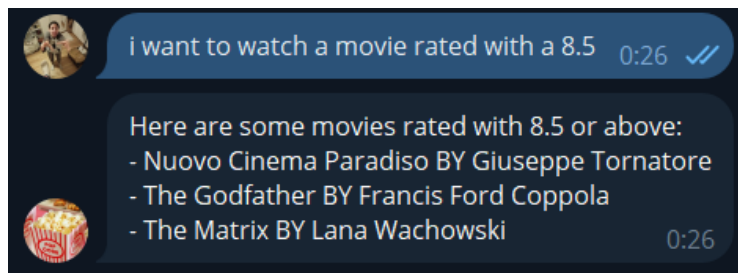


Figura 10: Búsqueda de película libre por puntuación en Telegram