Lampadaris Angelos

Special Topics in Language Technology: Multimodal and Dialogue Systems and Voice Assistants

Rasa chatbot: Movie Suggestions and Cinema Ticket Booking

Domain & Motivation

Domain:

Movies and cinema ticket booking - Easier access to useful information using natural language

Motivation:

- Sudden rise of streaming platforms
- Drop of cinema visitors
- Closing of video stores

Inspiration

This project is inspired by Cinobo; a movie theater that also works as a streaming platform offering easy access to a wide variety of movies and series from the ease of your own home, while also maintaining the movie theater which people can visit to watch movies the “old-fashioned” way!

Scenarios

Scenario 1: Similar Movies

Scenario 2: Movie Details

Scenario 3: Book a ticket

Scenario 4: Modify a ticket reservation

Scenario 5: Cancel a ticket reservation

Scenario 1: Similar Movies

Mockup Action: action_find_similar_movies

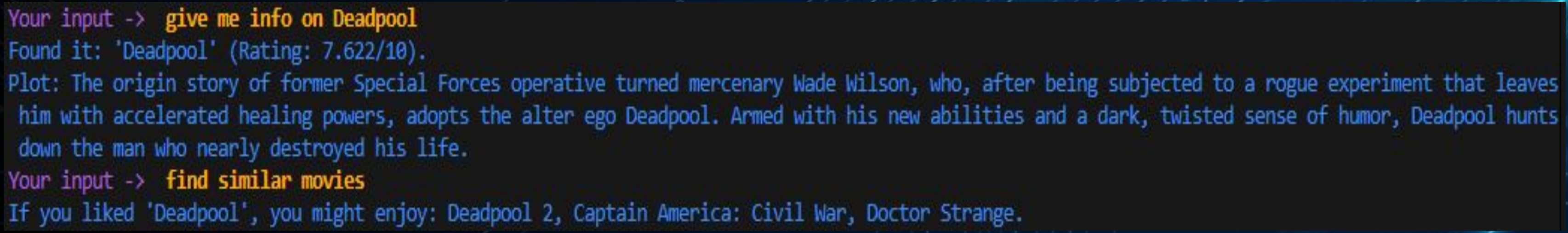
Functionality:
Given a movie (user input) bot returns 3 similar movies

Demonstrates:
Data drawn from API call

Usage:
Movie suggestions based on what the user like. More advanced than simple genre matching - a movie may belong in many genres.

Scenario 1: Similar Movies (& Scenario 2: Movie Details)

- User asks for information about a movie (Deadpool)
- Bot returns the information
- User asks for similar movies
- Bot carries over the slot and utterance “find similar movies” is valid as: “find similar movies to the movie ‘Deadpool’” - closer to Natural Language.



```
Your input ->  give me info on Deadpool
Found it: 'Deadpool' (Rating: 7.622/10).
Plot: The origin story of former Special Forces operative turned mercenary Wade Wilson, who, after being subjected to a rogue experiment that leaves
 him with accelerated healing powers, adopts the alter ego Deadpool. Armed with his new abilities and a dark, twisted sense of humor, Deadpool hunts
 down the man who nearly destroyed his life.
Your input ->  find similar movies
If you liked 'Deadpool', you might enjoy: Deadpool 2, Captain America: Civil War, Doctor Strange.
```

Scenario 2: Movie Details

Functionality:
User wants to obtain information about a movie

Demonstrates:

- Movie rating
- (short) summary of the movie
- Data provided from API

Important note:
Each movie title returned from this action is stored in a .txt file.
Usage: When patching the bot, new movie titles are added as a locally stored database (enhancing the ability to better distinguish the movies. Leaving room for further development)

Scenario 3: Book a ticket

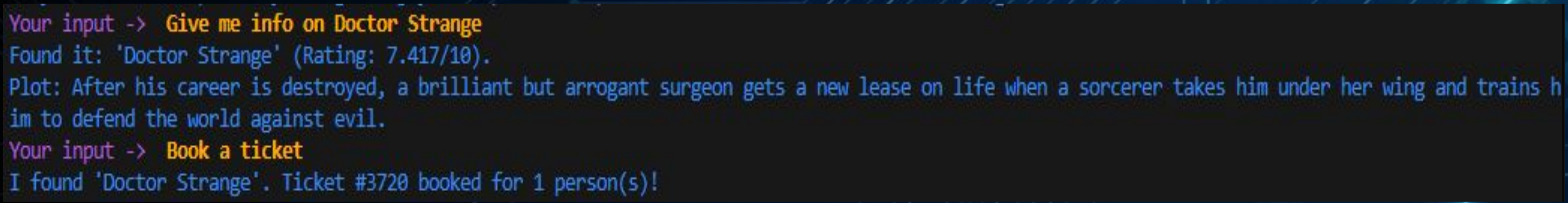
Functionality:
Provides a fast and easy way to reserve a seat (or more) to the movie theater using natural language. Motivates the user to visit the movies.

Demonstrates:
The booking id for each reservation, the number of tickets for the reservation and the movie the user selected.

Scenario 3: Book a ticket

Booking information is stored in Bookings.csv file with the columns: user_id, movie_title, booking_id, seats

User id is unique for each conversation (at this current version)

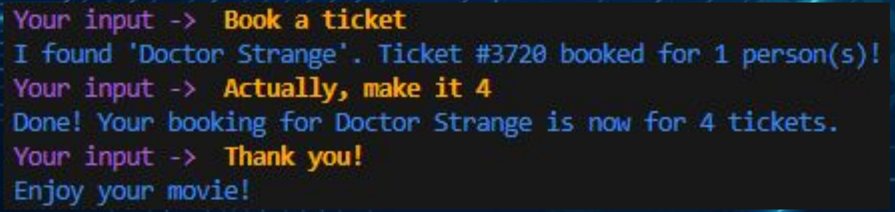


```
Your input ->  Give me info on Doctor Strange
Found it: 'Doctor Strange' (Rating: 7.417/10).
Plot: After his career is destroyed, a brilliant but arrogant surgeon gets a new lease on life when a sorcerer takes him under her wing and trains h
im to defend the world against evil.
Your input ->  Book a ticket
I found 'Doctor Strange'. Ticket #3720 booked for 1 person(s)!
```

Scenario 4: Modify a reservation

Functionality:
In case of a mistake during booking (either made from the user or the bot) or even a sudden change of plans.

Demonstrates:
The changes made in the reservation (change of number of tickets) and the movie that it is about.

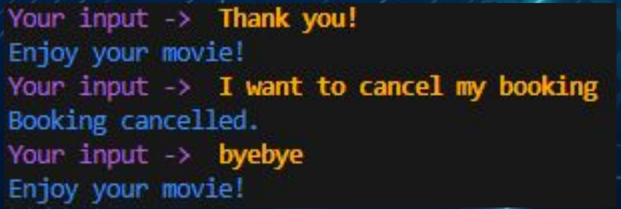


```
Your input ->  Book a ticket
I found 'Doctor Strange'. Ticket #3720 booked for 1 person(s)!
Your input ->  Actually, make it 4
Done! Your booking for Doctor Strange is now for 4 tickets.
Your input ->  Thank you!
Enjoy your movie!
```

Scenario 5: Cancel a reservation

Functionality:
If the user decides that he/she does not want (or is not able) to watch the movie, he/she gets to cancel the reservation.

Demonstrates:
A message confirming cancellation of the last reservation made from the same user.
Cancellations can be executed based on user id.



```
Your input ->  Thank you!
Enjoy your movie!
Your input ->  I want to cancel my booking
Booking cancelled.
Your input ->  byebye
Enjoy your movie!
```

Data integrations

The Movie DataBase (TMDB) API:

- Offers lists of movies with their corresponding plot summaries and ratings.
- Access with API key, generated from a simple account creation through the website.

Movies.txt

- A .txt file containing a list of movies available to search through TMDB. It gets updated with any new movie provided from the return of the "action_get_movie_details" action.

Data integrations

Movies.txt

- Works as a lookup table $\longrightarrow$ Used for integrating new data in every train.

Bookings.csv

- A file containing all the necessary details about each reservation.

Challenges and solutions

Numbers as text: The bot did not recognize numbers written in text (i.e. “six”) instead of an integer, in order to be aware of how many tickets to book.

Solution: A word mapping was implemented (i.e. one=1, two=2 etc) so that it can identify the right number. For instances such as “Book a ticket”, we simply included more examples in the nlu.yml, as well as other possible utterances which are suitable for bookings with more than one tickets.

Challenges and solutions

Movie titles: Some movie titles contain articles (“The Matrix”, “The Fantastic 4” etc) where slot is difficult to be recognized from the bot, or it fails to tell 2 movies apart (“The Matrix”, “Matrix”). Also, it faces challenges with synonyms or abbreviations of the title (“spiderman” instead of “Spider-Man”, “lotr” instead of “Lord of The Rings”).

Solution: To deal with ambiguity, the movies.txt is a temporary but working solution, in addition to explicitly adding more references of these movies as examples in the nlu.yml. Regarding abbreviations and synonyms, we added “synonyms” in nlu.yml so that it acquires the ability to correlate 2 or more slots that appear quite different between them.

Challenges and solutions

Confusion on actions to perform: There were instances where the bot was asked to provide information about a movie, but decided to search for a booking or make one. This problem showed that there were not enough instances of utterances correlated with each action or that they were overlapping.

Solution: Adding more examples helped the bot better distinguish the which was the right action to perform, while carefully examining all the examples given for a possible overlap.

Dialogue Policy

slots carry over: The most natural way to communicate is to assume that the last specific object (movie) referenced is the one that has to be booked or searched for similar movies.

Same goes for cancelation or modification of each booking (mostly due to low likelihood of booking more than one movie in one session).

Result: utterances such as *"book it"*, *"cancel my reservation"*, *"find similar movies"* etc, are considered valid for the current state of the bot.

Caution: this kind of assumptions should be handled more carefully in future updates.

Bot re-states the title: user is certain that the bot retrieved the right movie & booked the right number of tickets.

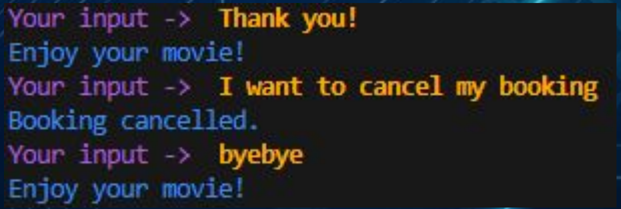
Limitations

- **Not enough instances for different utterances**

While the user has cancelled his booking, the bot responds: *"enjoy your movie"*.

Attempts were made to differentiate between two instances, but:

Increasing number and length of scenarios, appeared to be confusing for the model.



```
Your input ->  Thank you!
Enjoy your movie!
Your input ->  I want to cancel my booking
Booking cancelled.
Your input ->  byebye
Enjoy your movie!
```

Limitations

- **More data**

Movie titles and movies might appear with similar titles, either belonging in the same franchise or not. This was difficult to make it clear.

Temporary solution: increase of train data with movies.txt & more different instances of movies in nlu.yml.
(Also tried fuzzy matching, but it didn’t seem to do much)

PREPARE FOR THE FANTASTIC
FANTASTIC FOUR
Fantastic 4
O REED RICHARDS
H SUSAN STORM
O JOHNNY STORM
O BEN GRIMM

Future Improvements

- **Dialogue Policy:**
  - Past a certain threshold of uncertainty when slot filling, ask question: *“Do you mean [movie] with [actor]/[year of release]?”*
  - Tweaks on scenarios & intents for appropriate answer if no booking.
  - More friendly approach, but stick to the objective.

- **Minimizing API calls:**
  - Storing info such as movie details or movie similarities in local files while prioritizing info from local $\longrightarrow$ less calls.
  - Max calls = 40/second (hard to go public!!!)
    - (Since API responds quickly, add message *“searching”/”thinking”* with sleep to control traffic).

Future Improvements

- **Seats/room choice**
  - When booking, user can ask for seats *“close to the center/back/front of the screening room”*
  - Bot might suggest a more suitable screening room based on the main genre of the movie (i.e. 3d rooms for sci-fi movies).
  - Improve booking experience making it time and date specific (tool like Duckling).

- **Movie theater sales**
  - Suggest food/drinks sold by the movie theater. Making a purchase through the bot, helps the user skip the line when visiting the movies.

Conclusion

The chatbot utilizes real-world data about movies. It assists the user handling the full booking experience while being able to respond in a more natural manner (understands pronouns: *“Book it”*).

Using TMDB API data such as movie plots, ratings and similarity are provided, which help the user discover which movie to watch with low latency calls.

All in all, the chatbot can be a great assistant unifying real-word data with offline data, providing a natural conversation and helping efficiently all sorts of movie enthusiasts.